For Reference

NOT TO BE TAKEN FROM THIS ROOM

Ex libris
universitatis
albertaensis

QUAECUMQUE VERA

Digitized by the Internet Archive
in 2023 with funding from
University of Alberta Library

https://archive.org/details/Woo1972

THE UNIVERSITY OF ALBERTA

AN O.R. APPROACH TO OPTIMIZE COMPUTER FILE SYSTEMS

by

Ⓒ PETER WOO

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

FALL, 1972

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled AN O.R. APPROACH TO OPTIMIZE COMPUTER FILE SYSTEMS submitted by Peter Woo in partial fulfillment of the requirements for the degree of Master of Science.

ABSTRACT

An operations research approach is proposed for the optimal allocation of input/output buffer memory among files stored on various types of auxiliary mass storage devices. The objectives are to minimize the input/output time and storage requirements of computer file systems. A mathematical model of a file subsystem is constructed with a discrete non-linear objective function and discrete linear constraints. Operations research techniques such as Lagrange multiplier method and non-linear programming are discussed.

An extension of the problem from a magnetic tape file system to a file system resident on direct access storage devices is introduced. Implementation problems are briefly mentioned. Illustrative numerical examples with sensitivity analysis are included in the appendices.

ACKNOWLEDGEMENTS

I wish to thank my supervisor Prof. W.S. Adams for his guidance and patience, and the Department of Computing Science for the financial assistance, with which this project was undertaken. I wish also to thank my colleague Mr. B.A. Reid for the use of his non-linear program.

TABLE OF CONTENTS

Page

CHAPTER I - INTRODUCTION 1

1.1 Resource Allocation in Computer Systems 1
1.2 Literature Survey 2

CHAPTER II - FRAME OF REFERENCE 5

2.1 An Hierarchical Approach 5
2.2 System Description 5
    2.2.1 Hardware Configuration 5
        2.2.1.1 Hardware Parameters 7
    2.2.2 Software Support 8
        2.2.2.1 Software Parameters 8

CHAPTER III - IDENTIFICATION OF PROBLEM 9

3.1 Effects of Blocking on Tape File Systems 9
    3.1.1 Effects on I/O Time and Tape Length 15
        3.1.1.1 Reduction of I/O Time & Overhead 16
        3.1.1.2 Tape Cost Savings 17

CHAPTER IV - TREATMENT OF PROBLEM 19

4.1 Methodology of Treatment 19

Page

4.1.1 Principles of Optimization 19

4.1.2 Local Optimization of a Single Program. 20

4.1.2.1 Lagrange Multiplier Method ... 22

4.1.3 Global Optimization of Entire System .. 24

4.2 Formulation of the Objective Function 27

4.2.1 Selection of Objectives 29

4.2.1.1 Cost Criteria 29

4.2.1.2 Throughput Criteria 29

4.3 Formulation of the Memory Constraints Matrix.. 31

4.3.1 Effects of Manipulating a_{ij} 32

4.3.2 Effects of Varying M_i 32

4.4 Important Considerations in System Analysis .. 33

4.4.1 Essential Information 34

4.4.1.1 Hardware Configuration 34

4.4.1.2 Software Support 35

4.4.1.3 Master System Flowchart 35

4.4.1.4 System Audit Data 36

4.4.1.5 File Characteristics 36

4.4.1.6 Sort/Merge Utilities 37

4.4.2 Desirable Information 37

4.4.2.1 Plans for Future Growth 37

4.4.2.2 Peak Loads 38

4.4.2.3 System Bottlenecks 39

4.4.2.4 File Activities 43

4.4.2.5 Statistical I/O Error Rates .. 44

4.4.2.6 Selection Criteria for Blocking Factors 44

Page

CHAPTER V - EXTENSION OF PROBLEM 45

5.1 Direct Access Storage Devices 45
5.2 Disk File Organization 45
    5.2.1 Sequential Organization 45
    5.2.2 Partitioned Organization 46
    5.2.3 Indexed Sequential Organization 46
    5.2.4 Direct Organization 46
5.3 Disk Storage Organization 47
    5.3.1 Track Formats 47
    5.3.2 Record Formats 47
5.4 Effects of Blocking on DASD File Systems 49
    5.4.1 Effects on File Compaction 50
    5.4.2 Effects on I/O Time 52
5.5 Isolation of Problem 53
5.6 Methodology of Treatment 54
    5.6.1 Formulation of Objective Function 54
    5.6.2 Formulation of Memory Constraints Matrix 55
    5.6.3 Discrete Optimization Methods 56

CHAPTER VI - IMPLEMENTATION 57

6.1 Research Recommendations and Managerial Action 57
    6.1.1 Rational Rejection 57
    6.1.2 Resistance 58

Page

6.1.3 Acceptance 59

6.1.4 Implementation 59

6.1.4.1 Sustained Implementation 59

6.1.4.2 Autonomous Implementation ... 60

6.2 System Performance Monitoring 60

CHAPTER VII - CONCLUDING REMARKS 61

7.1 Economic Payoffs and Intangible Benefits 61

7.2 Applicability in Future Technology 62

APPENDIX A - LOCAL OPTIMIZATION BY LAGRANGE MULTIPLIER METHOD 63

APPENDIX B - GLOBAL OPTIMIZATION BY GRADIENT PROJECTION METHOD 68

BIBLIOGRAPHY 92

LIST OF FIGURES

Page

Fig. 2.1 Hardware configuration of a typical commercial data processing centre 6

Fig. 3.1 Effect of blocking on tape device 10

Fig. 3.2 Tape I/O time per file as function of B.F. ... 11

Fig. 3.3 Tape length required per file as function of B.F. 12

Fig. 3.4 Tape I/O time per file as function of B.F. (logarithmic plot) 13

Fig. 3.5 Tape length required per file as function of B.F. (logarithmic plot) 14

Fig. 4.1 A program using n tape files 20

Fig. 4.2 A subsystem of three interrelated programs and their associated files 24

Fig. 5.1 DASD track formats (IBM) 48

Fig. 5.2 DASD record formats 49

Fig. 5.3 No. of tracks required per file as function of B.F. (IBM/2314 disk, with keys) 51

Fig. A-1 An application program using four tape files.. 63

Fig. B-1 A subsystem of 3 interrelated programs and 6 tape files 68

LIST OF TABLES

Page

Table 1 - 5 IBM/2400-series magnetic tape unit characteristics 92

Table 6 - 10 IBM/2300-series DASD characteristics ... 99

Table 11 IBM/7320 drum storage characteristics ... 105

CHAPTER I

INTRODUCTION

1.1 Resource Allocation in Computer Systems

The use of operations research techniques to solve problems of allocation of large corporate resources is relatively well established. [60] However, the application of these techniques to optimize resource allocation in computer systems has not yet been widely reported. The latter situation is due mainly to the fact that computer systems hitherto have not been a large corporate resource involving multi-million dollar capital or operating expenditures that traditionally occupy the attention of operations researchers.

This situation is changing, however, as systems become more complex and expensive. Consequently, the need for a more rational and systematic approach to this problem is becoming increasingly apparent. A foremost requirement for a fruitful application of these powerful techniques is the correct identification of problem areas in computer systems. Such problems may exist in the file design, job scheduling or hardware configuration of a system.

After a brief literature survey in the next section, this thesis will concern itself with the identification and analysis of one such problem area, followed by a discussion of possible treatments and ramifications of the problem.

The appendices contain detailed numerical examples worked out for a small subsystem to demonstrate the use of the proposed methods of treatment. Computer program listings (in FORTRAN) are included for a complete documentation for the purpose of implementation of the proposed project.

1.2 Literature Survey

The published works referenced in this thesis fall into two broad categories, namely OR techniques and computer systems performance.

There is not a great deal of published work on the application of OR (constrained minimization/maximization) techniques to the problem of optimal allocation of computer systems resources. Chu [9] discusses the optimal file allocation in a multiple computer system. He treats the problem of optimal sharing of files in a computer network. Walker [32] and Waters [33] both present basically the same approach to optimize blocking factors on magnetic tape files using the Lagrange multiplier method. However, their treatments are limited to tape files used in an individual program. Woodrum [36] develops an analytic queueing model of floating buffers to minimize program run time. His model is a semi-Markovian process. He also makes a comparison between the performance of floating buffering and the usual double buffering. Silver, et al [29] have developed a quantitative rule for automatic allocation of systems resources in a multiprogramming environment.

On the effects of component interactions on systems performance, Hellerman (19) analyzes the effect of overlapping processes on throughput in an idealized multiprogramming environment, and Chang and Wong (7) discuss channel interference. File organization methods and their performance are discussed by Abate et al (1), Chapin (8), Collmeyer (11), and Lowe (22).

Many other studies include various aspects of computer systems performance optimization, but are only marginally related to the use of OR techniques in constrained minimization/maximization problems. They generally fall into the categories of queueing analysis and dynamic simulation.

In the field of OR techniques and algorithms, on the other hand, there is a wealth of published works on constrained minimization/maximization problems. Only a subset of the most relevant works are cited in the bibliography. Of immediate interest to this thesis are Rosen's gradient projection algorithm (82) and Everett's treatment of a generalized Lagrange multiplier method (46). Other key references include Dorn's extensive survey of non-linear programming methods (72), textbook treatments of non-linear programming by Abadie (62), Hadley (77), Kunzi et al (79), Mangasarian (80), and Zangwill (87). Book treatments of integer programming include those by Hu (100), Muth and

Thompson (105), Tonge (108), Wagner (61), and Weingartner (110). Survey papers on integer programming methods include those by Balinski (90) and Beale (91). The problem of sub-optimization is examined by Hitch (50). The whole question of OR implementation is fully discussed in a book on this subject by Huysmans (52).

This literature survey on computer systems optimization and OR methods is necessarily sketchy, and the reader is referred to a more comprehensive bibliography at the back.

CHAPTER II

FRAME OF REFERENCE

2.1 An Hierarchical Approach

In his paper on the design of the "THE"-multiprogramming system, Dijkstra [14] introduced the concept of hierarchical testing of a complex system during its construction. He claimed that a system is soundly constructed if it is thoroughly tested for all possible eventualities at each hierarchy of increasing complexity. This hierarchical approach will be adopted here in establishing a frame of reference in which system performance can be predicted with some measure of confidence.

2.2 System Description

2.2.1 Hardware Configuration

In general, the hardware configuration of a typical commercial data processing installation comprises a central processing unit (CPU), several core boxes of main memory, one or more selector channels, a multiplexor channel, several magnetic tape devices, disk devices, and other sundry peripheral devices.

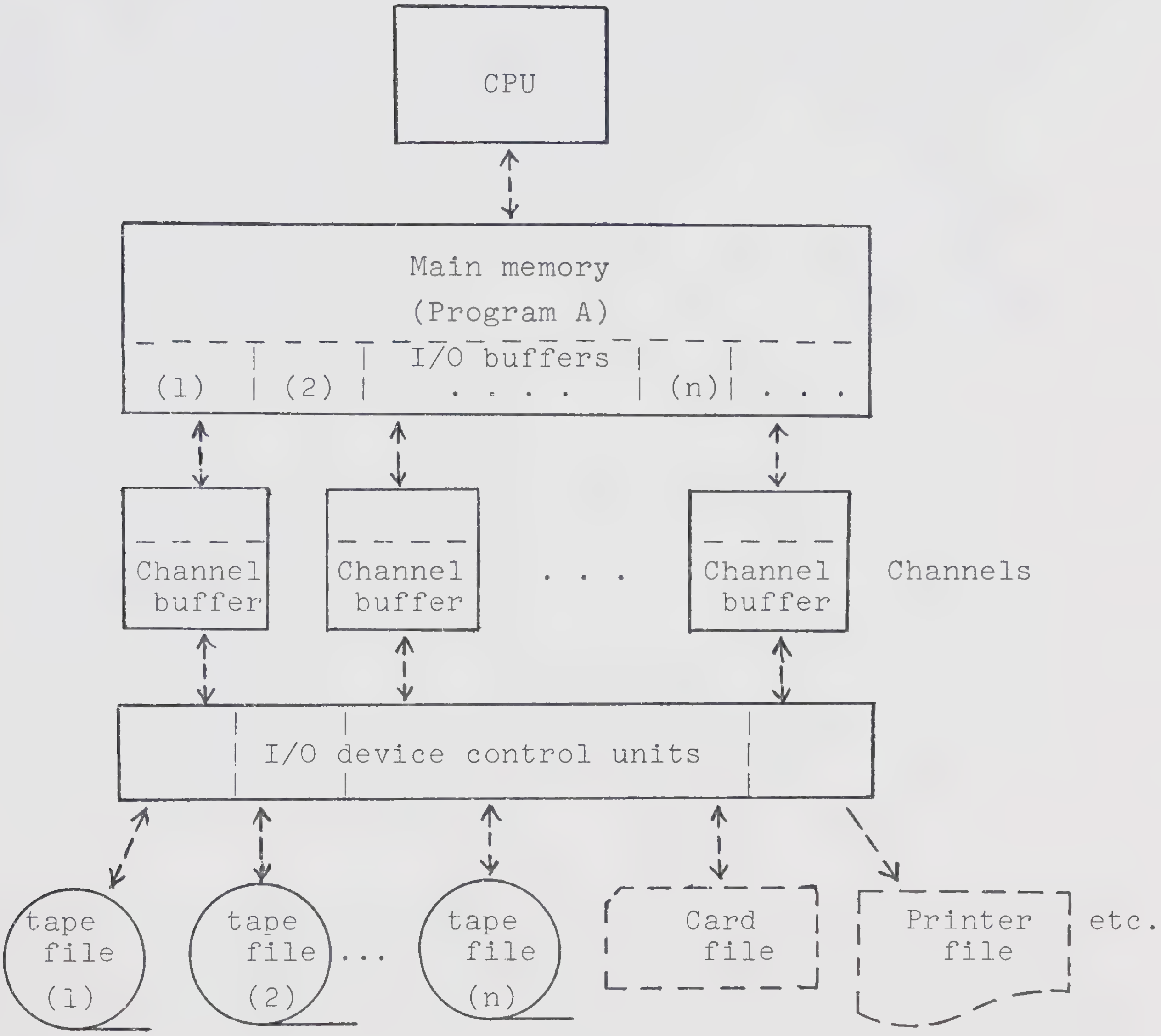


Fig. 2.1 Hardware configuration of a typical commercial data processing center.

To a large extent, the input/output (I/O) control units determine the characteristic manners in which the different types of files are accessed. The unit-record files (e.g. card reader and line printer) can only be accessed in single units of fixed size records. Thus there is no I/O parameter pertaining to these files that can be easily manipulated to improve the system performance. Furthermore, these unit-record files and devices are directed to the multiplexor channel, which is structurally and functionally unconnected to the high-speed peripheral devices such as disk and tape. That is, there is no mutual channel interference or memory contention as such between the high-speed and low-speed peripheral devices. In light of this functional separation, one can reasonably isolate and remove that part of the system and its requirements associated with the low-speed devices from an overall system consideration for the purpose of optimizing file design and system performance.

2.2.1.1 Hardware Parameters

Each hardware device normally performs with a characteristic speed and load capacity. These constitute the primary hardware parameters of the system. The ones which are of interest in the present context are the channel buffer capacity,[1] the recording densities and transfer rates of

---

[1] The channel buffer capacity may be considered to be the longest chain of data that can be transferred by one I/O command. (E.g. on the IBM/360 it is 32 KB.)

tape and disk devices, the rotational and seek times of disk devices, and the inter-block gap length and start-stop time of tape devices. Other secondary parameters such as I/O error recovery rates will be considered, but will not be incorporated into the mathematical model of the system.

2.2.2 Software Support

In order to focus our attention on the identification of the problem and observe the primary effects of the proposed treatment, let us first consider the simplest operating system -- a single-stream batch processing environment. Apart from the necessary core-resident control programs, there is only one application program residing in the entire remaining core memory. It is executed until job completion before the next job is brought into core memory. That is, there is no overlap of CPU and I/O operations between different programs.

All tape files will have fixed size records that can be blocked. Disk files will not be considered in this system initially, since they involve many more parameters due to the different methods of organization and access.

2.2.2.1 Software Parameters

The partition sizes of main memory can be fixed at system generation time, or dynamically variable according to users' requirements. The maximum number of alternating I/O buffers per file is determined by the I/O control program.

CHAPTER III

IDENTIFICATION OF PROBLEM

3.1 Effects of Blocking on Tape File Systems

In a typical data processing installation of a firm, the majority of regular production jobs involve magnetic tape files. These files, by the nature of their recording medium and access mechanism, lend themselves to the various sequential access methods.

It is an elementary fact in data processing that such files can be read or written in a shorter time[2] if the individual records are grouped together in blocks of physically contiguous records, separated by inter-block gaps (IBG's). This physical blocking of records reduces the total number of IBG's on a file. Thus the larger the files are, the more substantial will be the tape saving due to file compaction.

Figure 3.1 shows how blocking affects the relative amount of tape wasted compared to the length of tape actually containing records. It is quite apparent that the bulk of the tape contains empty IBG's when the file is unblocked (i.e. blocking factor = 1). When blocked by a factor of two (BF= 2), there is still considerable tape wastage. However, it should be noted from the graphs in figures 3.2 and 3.3 that the change of BF from 1 to 2 yields the greatest percentage reduction (almost 50%) in total tape length requirement and I/O time per file.

[2] The time referred to here is the total I/O time (including overhead) rather than the total channel transfer time, which is independent of the blocking factor used.

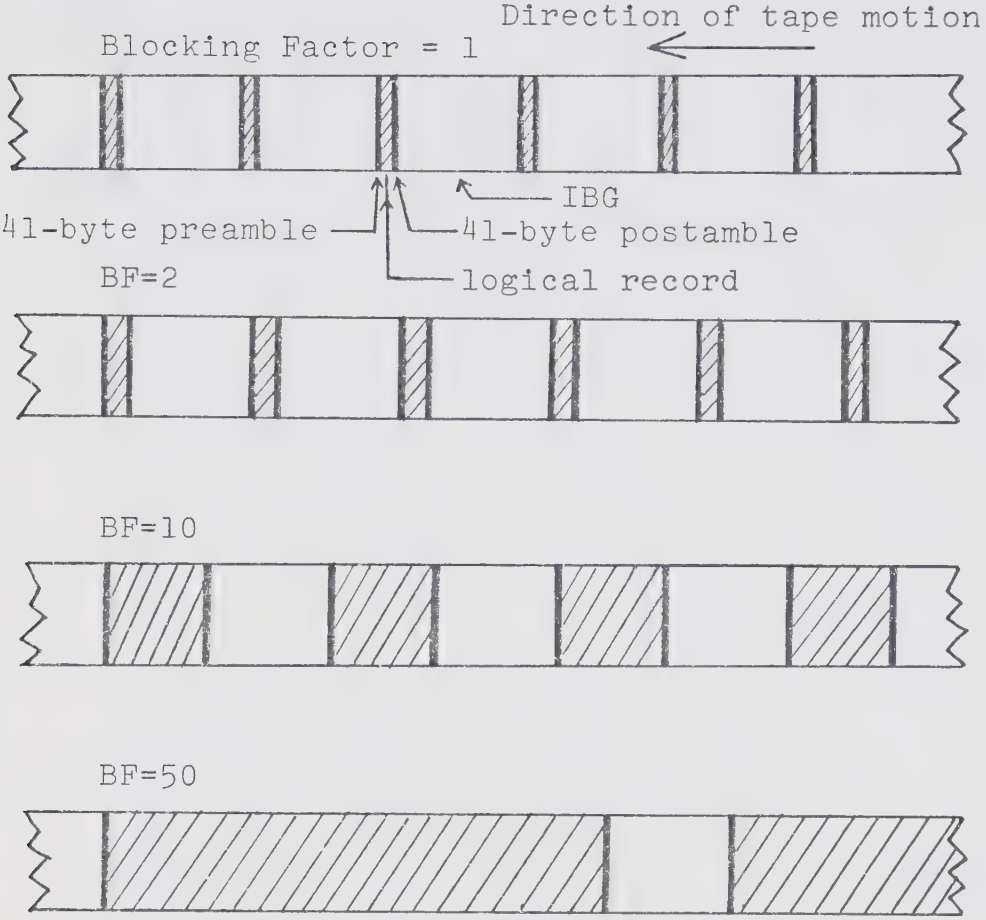


Fig. 3.1 Effect of blocking (80-byte logical records) on IBM/2415 models 4-6 tape device, 9-track, 1600 bpi, no overlap. (Actual size)

Theoretically, for files that can be blocked, the entire file should comprise one block, allowing it to be processed with only one I/O command. In practice, however, this is seldom feasible or desirable.[3] Hence, some compromise has to be reached in determining the optimal blocking factor subject to the given constraints.

[3] In practice, the size constraints of main memory and channel buffer capacity impose an upper bound well below the size of most files. Furthermore, statistical I/O error rates involving reread/rewrite of entire blocks become a significant consideration as the block size becomes too large.

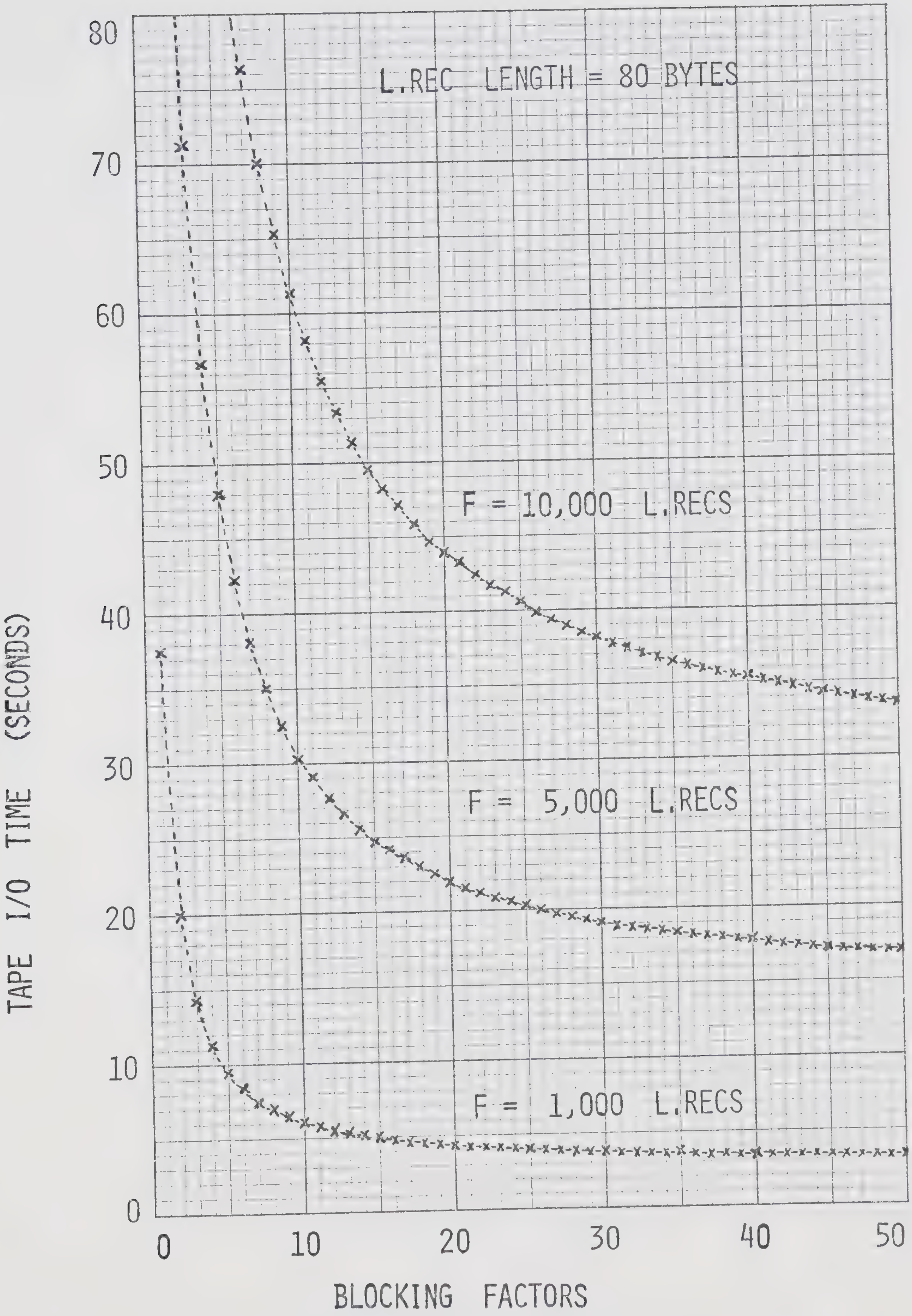


FIG. 3.2 TAPE I/O TIME PER FILE AS FUNCTION OF B.F.
(IBM/2415 MOD 4-6 TAPE DEVICE, 9-T, 1600 BPI, NO OVERLAP)

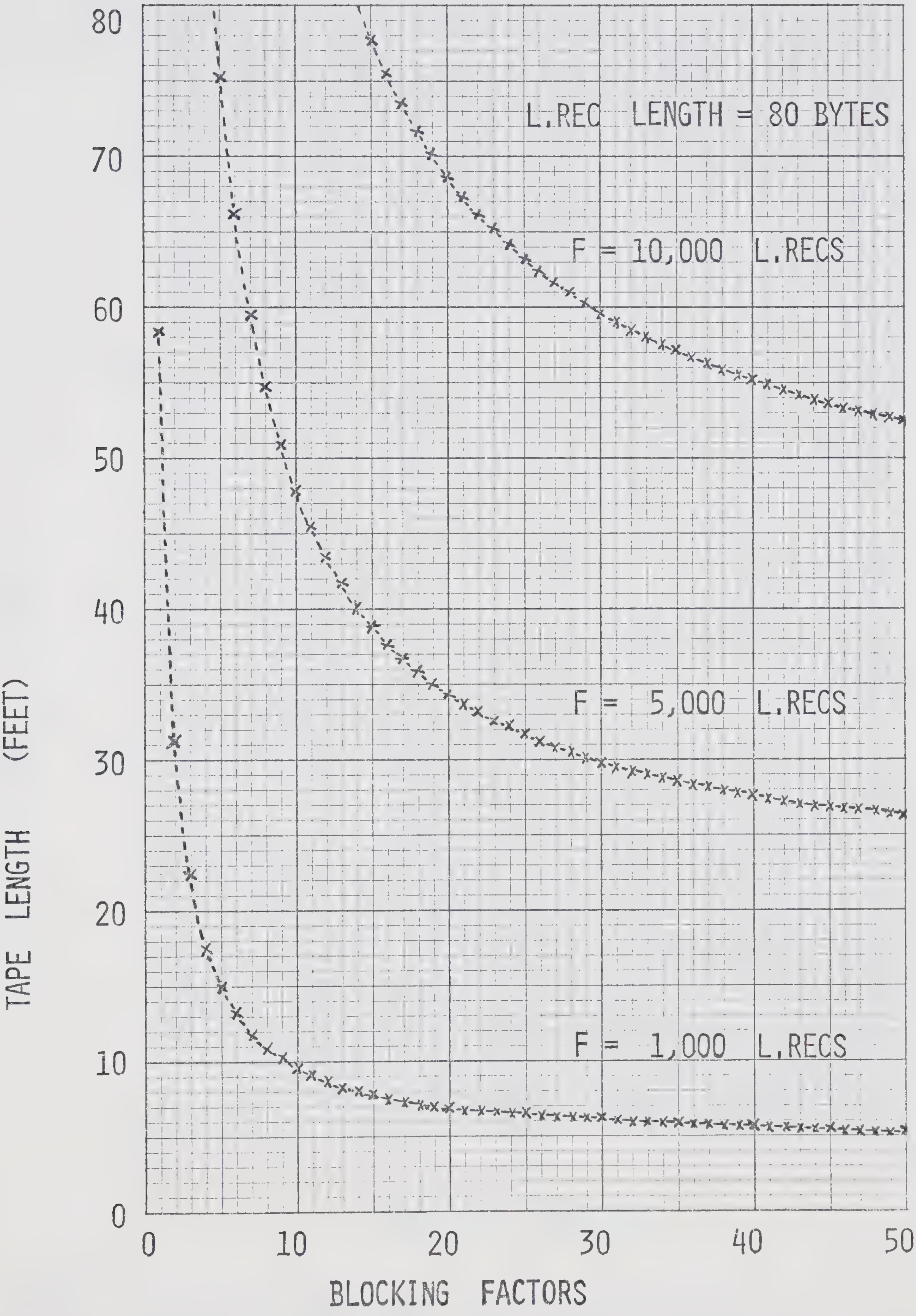


FIG. 3.3 TAPE LENGTH REQUIRED PER FILE AS FUNCTION OF B.F.
(IBM/2415 MOD 4-6, TAPE DEVICE, 9-T, 1600 BPI, NO OVERLAP)

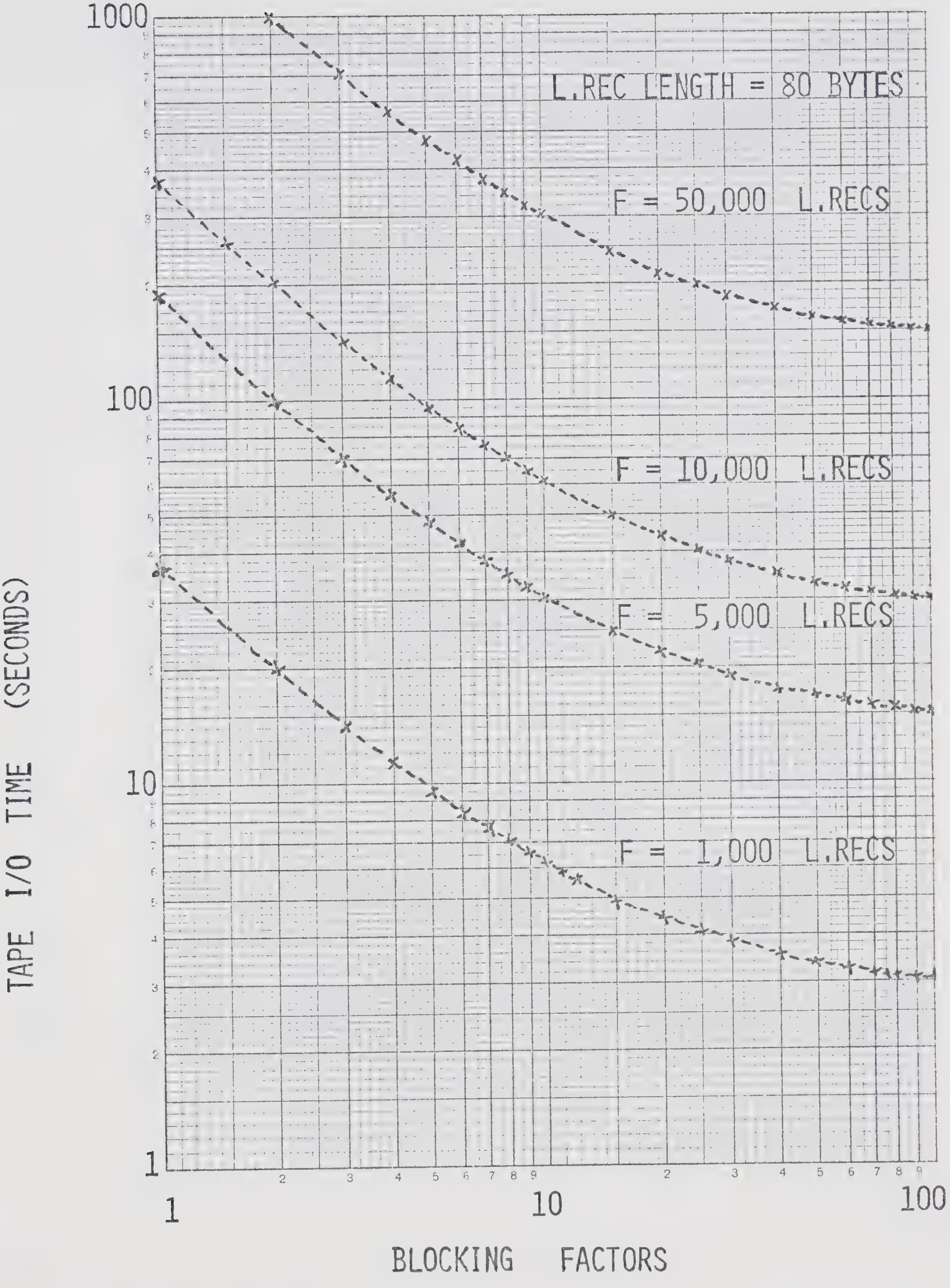


FIG. 3.4 TAPE I/O TIME PER FILE AS FUNCTION OF B.F.
(IBM/2415 MOD 4-6 TAPE DEVICE, 9-T, 1600 BPI, NO OVERLAP)

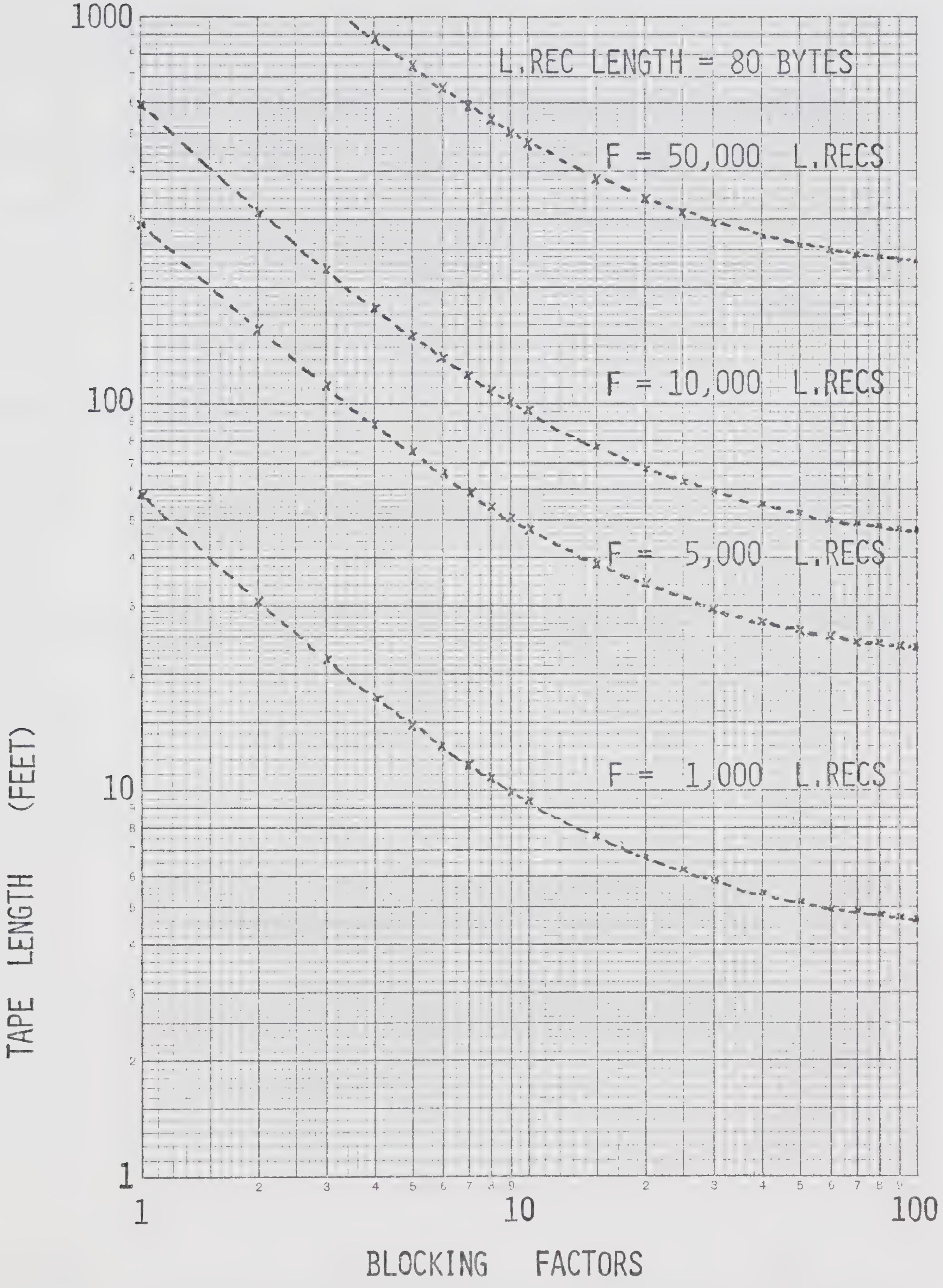


FIG. 3.5 TAPE LENGTH REQUIRED PER FILE AS FUNCTION OF B.F.
(IBM/2415 MOD 4-6 TAPE DEVICE, 9-T, 1600 BPI, NO OVERLAP)

3.1.1 Effects on I/O Time and Tape Length

In general, the inter-block start-stop time per block is independent of the block size.[4] Thus the time required to read/write a block is given by

$$t = S + C \times B \qquad (3.1)$$

Hence, the read/write time per file is given by

$$T = K \times (S + C \times B) \qquad (3.2)$$

Similarly, the total tape length required per file is given by

$$L = K \times (G + D \times B) \qquad (3.3)$$

where t = I/O time per block
T = I/O time per file
L = Tape length per file
D = Tape length per character
C = I/O time per character
S = Inter-block start-stop time
G = Inter-block gap length
K = No. of blocks per file
B = No. of characters per block

---

[4] This is a basic assumption in our problem identification. In fact, it varies over a range which depends on the manufacturer and the model of the tape device, as well as the file organization and the processing time of the program. In some instances, the tape movement need not come to a complete stop in between blocks. Here, the mean start-stop time is used.

From equations 3.2 and 3.3, the total I/O time T and tape length L can be determined with respect to the block size B.

3.1.1.1 Reduction of I/O Time and Overhead

From figure 3.2 one can readily observe that the most significant variations in I/O time lie in the range of blocking factors between 1 and 10. It is also evident from the family of curves in figure 3.4 that for any particular difference in blocking factors, the absolute variation in I/O time is directly proportional to the file size, whereas the percentage variation is independent of the file size.

The reduction of the number of IBG's on a tape file yields primarily a reduction of the total inter-block start-stop time. Concomitant with this is the reduction of I/O overhead, which consists of I/O commands and supervisory routines for handling I/O interrupts (and program switching/ swapping in multiprogramming/time-sharing systems). The latter are secondary or tertiary effects with which we shall not be concerned in our basic frame of reference. These effects will be discussed more fully in a more complex frame of reference at a later stage.

3.1.1.2 Tape Cost Savings

It is well worth noting from figure 3.5 that most transaction files of less than about 10,000 records (of 80-byte logical record length), when blocked, can easily be stored on mini-reels of 300-400 feet long, as compared to full reels of 1200-2400 feet long. When applied to the entire DP system of a large corporation which uses 1,000 to 5,000 reels of tape, this can represent a substantial saving in the cost of tape acquisitions for an installation.

Two basic approaches can be taken to reduce the tape requirements. One can store all small files on separate mini-reels, or combine several small files into one multi-volume file on a full reel. Care should be taken to use the latter approach in conjunction with the job scheduling, so that the small files for consecutive jobs are stored on the same reel in order to minimize the operator's tape set-up time. In a single-stream batch processing environment, usually found in smaller installations, an excessive set-up time can become an important factor in degrading the throughput of the system. Hence the mini-reel approach might put such a system at a disadvantage. In a larger multiprogramming environment, however, the set-up time can usually be imbedded in the processing time of other jobs through internal scheduling (such as HASP), by which the operator is notified of set-up requirements of impending jobs in advance on the keyboard console. In such a situation, the mini-reel approach

would provide more flexibility of scheduling without suffering the above-mentioned drawback. Normally, a combination of these two basic approaches would be adopted in order to make full use of the characteristics of each application subsystem.

CHAPTER IV

TREATMENT OF PROBLEM

4.1 Methodology of Treatment

4.1.1 Principles of Optimization

Optimization of tape I/O time and length requirement can be achieved through the mathematical determination of blocking factors, using the calculus of variations and non-linear programming methods. In principle, one tries to minimize (or maximize) an objective function subject to certain system constraints.

For a program using two or more tape files, a certain upper bound on the sizes of the I/O buffers is imposed by system constraints, (e.g. job partition size, program size, channel buffer capacity, statistical I/O error rate, etc.), dependent on what hardware configuration is used, (e.g. size of main memory, number of selector channels, etc.), and what software support is available, (e.g. single-stream batch, multiprogramming, etc.).

This upper bound limits both the individual I/O buffer sizes and their aggregate size. The objective is to find an optimal arrangement of blocking factors which would minimize the total number of IBG's on the tape files subject to this buffer memory constraint.

One might ask at this point whether the minimization of the number of IBG's is the most suitable criterion to choose

for optimizing the system performance. The choice of an objective is a crucial part of the correct identification and treatment of the problem. Accordingly, this question will be dealt with more fully in section 4.2 on the formulation of the objective function.

4.1.2 Local Optimization of a Single Program

Let us consider the general case of a program involving n tape files.

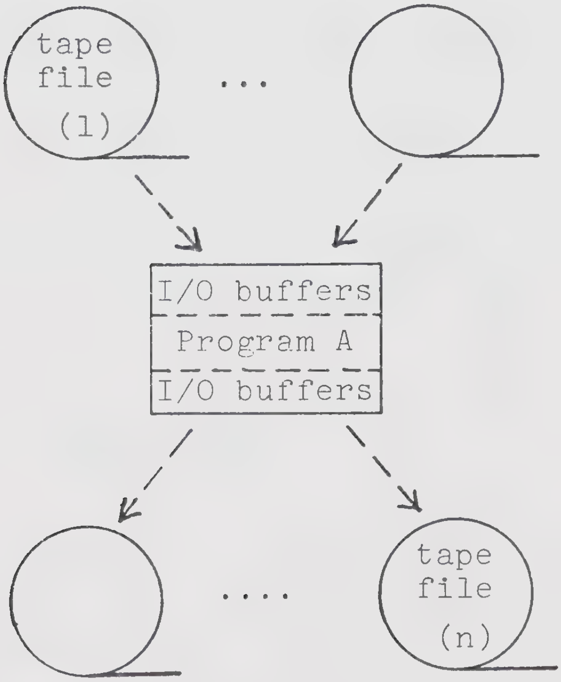


Fig. 4.1 A program using n tape files.

To optimize the I/O time and tape length requirement, we minimize the non-linear objective function [5]

$$z = \frac{F_1}{X_1} + \quad \dots \quad + \frac{F_n}{X_n} \qquad (4.1)$$

subject to the linear memory constraint equation

$$a_1 R_1 X_1 + \quad \dots \quad + a_n R_n X_n + S = M \qquad (4.2)$$

$$X_j > 0, \qquad j=1,2, \dots ,n$$

where z = Total no. of blocks (IBG's) processed per job

M = Total core memory available for tape buffers

X_j = Blocking factor for file j (records per block)

F_j = Size of file j (records per file)

a_j = No. of alternate tape buffers per file j

R_j = Record size for file j (characters per record)

S = Slack variable

In equations 4.1 and 4.2, the parameters a_j and M can be incrementally varied to observe their effects on the solutions for the X_j's. Various techniques can be used

---

[5] This model can be reformulated to consist of a linear objective function subject to a non-linear constraint. But, in general, the former model is easier to solve with the available computer algorithms.

to find the desired solutions. The mathematical techniques applied to solve the model can be greatly simplified by observing the fact that the discrete objective function z is a simple, well-behaved function, which can be treated as a smooth, continuous function with no singularity within the feasible region. For simplicity of approach, the method of Lagrange multiplier [46,57] can easily yield accurate results by hand calculations or with the use of a short program.

4.1.2.1 Lagrange Multiplier Method

Taking partial derivatives of z and M in equations 4.1 and 4.2 with respect to X_j and equating to zero for all j, we obtain the ordinary partial differential equations

$$\frac{\partial z}{\partial X_j} + \lambda \frac{\partial M}{\partial X_j} = 0, \qquad j=1,2, \ldots ,n \qquad (4.3)$$

where λ is the Lagrange multiplier.[6]

---

[6] λ assumes no physical significance in our solution, as we are only interested in solving for the X_j's, the blocking factors. However, an important economic interpretation is attached to λ, which represents the incremental reduction in the total number of blocks processed when the total available buffer memory is incremented.

From (4.3) we obtain a set of n homogeneous simultaneous algebraic equations in $n+1$ unknowns

$$-\frac{F_j}{X_j^2} + a_j R_j \lambda = 0, \qquad j=1,2, \ldots ,n \qquad (4.4)$$

Combining (4.4) with the original memory constraint equation, we can thus solve algebraically for all the X_j's. The file size F_j and record size R_j are known constants for each file,[7] and a_j and M are decision variables.

It should be remembered that the objective function and variables in the physical system can only take on positive integer values. The mathematically obtained optimal solution can then be rounded off heuristically and tested for feasibility and optimality to give a physically optimal or near optimal solution. The unused portion of the available buffer memory after a solution is rounded off is simply taken up by the slack variable S.

[7] In general, the file size F_j tends to vary over a period of time, in some cases considerably. Under those circumstances, F_j can only be approximated by its mean size.

4.1.3 Global Optimization of Entire System

When the application programs involving tape files in a data processing department are considered as an integrated system, in which some files are interrelated through more than one program, the optimization problem takes on a new dimension of complexity.

Here we shall define a 'shared file' as one which serves as input to more than one program, and a 'transmitted file' as one which is the output from one program and the input to another program.

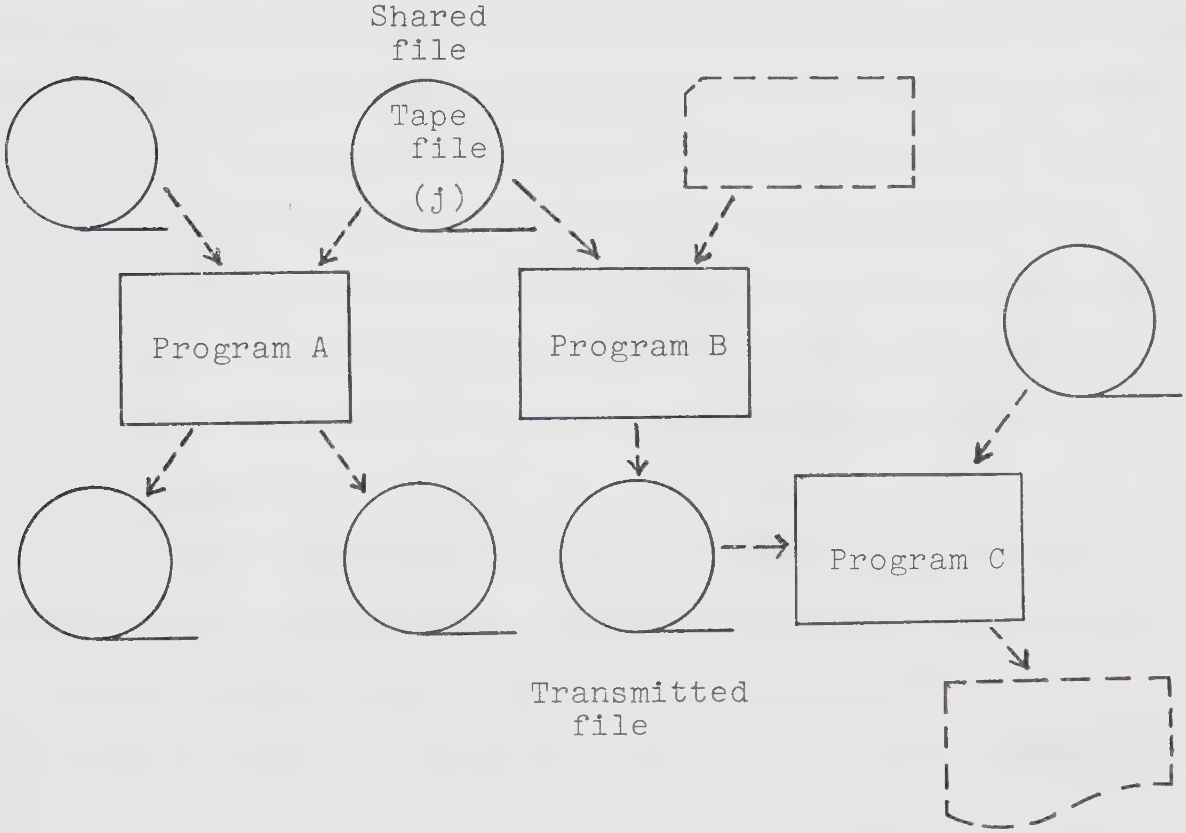


Fig. 4.2 A subsystem of 3 interrelated programs and associated files.

Consider the problem this way: A tape file j is shared between programs A and B. If its blocking factor is calculated subject to the memory constraint equation associated with program A, (local optimization), it is assigned a value X_{Aj}. Similarly, using the memory constraint equation for program B, X_{Bj} is obtained. Since it is in fact the same file being considered, it must necessarily be assigned a unique blocking factor X_j, calculated subject to both the original two constraints, which become inequalities (less than or equal) in order to yield a unique solution. It is important to realize that in a complex system, the optimal X_j need not necessarily be X_{Aj} or X_{Bj} or their average, and an arbitrary assignment of X_j would entail readjustment of the BF's for all the tape files in the subsystem. Often, such a perturbation of a complex system is sufficient to induce a significant deviation of system performance from the optimal. This is why optimization techniques such as non-linear programming are necessary to ensure a truly optimal solution.

Let us consider the general case of a subsystem of m interrelated application programs involving n tape files. The form of the objective function z remains the same as for a single program.[8] For each program, there is an inequality

---

[8] The objective function now applies to the entire subsystem as a whole, although, for the same number of tape files, it takes the same form as in the case of a local optimization of a single program.

memory constraint. Thus we minimize z subject to a set of m memory constraints.[9]

$$\begin{array}{ll} a_{11}R_1X_1 + \ \dots \ + a_{1n}R_nX_n + S_1 \leq M_1 & \\ \vdots \qquad\qquad\qquad\qquad \vdots & \qquad (4.5) \\ a_{m1}R_1X_1 + \ \dots \ + a_{mn}R_nX_n + S_m \leq M_m & \\ X_j > 0, \qquad j=1,2, \dots ,n & \end{array}$$

where a_{ij}, R_j, X_j and S_i have the same meaning as in the case of a single program. Here M_i denotes the maximum available tape buffer memory per program i.

The solution to this problem can be found by applying certain non-linear programming techniques, such as the Gradient Projection Method.[82, 83] These techniques have been summarized in a paper by Dorn [72], who surveys the developments in this field up to the early sixties.
A theoretical exposition of these methods would be beyond the scope of this application-oriented dissertation, and hence

---

[9] Some of the coefficients of X_j, i.e., some elements of the matrix $(aR)_{ij}$, will be zero. The physical interpretation of this is that not all the files will be used in any particular program. Thus in general, the larger a subsystem is, the more sparse will be the memory constraints matrix. In light of this characteristic, algorithms that are specially well-adapted to solving large sparse matrices (of the order of say 50 × 30) can be employed when dealing with large subsystems, (e.g. with 20 programs and 30 tape files).

will not be attempted. Suffice it to say that given a program which implements a suitable algorithm, a person trained in such techniques can easily interpret the numerical results. However, to set up a mathematical model to represent this problem would require a skilled analyst of computer systems to evaluate the relative significance of numerous interrelated parameters.

4.2 Formulation of the Objective Function

The objective function z can be considered as the total number of tape blocks (IBG's) processed (read/write) by the entire subsystem in one run cycle. A subsystem is defined here as a group of programs (and their associated files) in which each program is related to at least one other program in the group through a straddled file (i.e. shared or transmitted files collectively). In our present context, a run cycle of a subsystem is defined as the run cycle of the least frequently run program in the group.

If a tape file is straddled, say, over programs A and B, and A is run daily (5-day week) whereas B is run weekly, then A should have a relative weight of 5 to 1 in totalling up the number of tape blocks processed in that period. Thus to reflect the relative weight (frequency) of the number of blocks processed by each program with respect to each file, the general equation 4.1 should be modified to

$$z = f_1 \frac{F_1}{X_1} + \ \ldots \ + f_n \frac{F_n}{X_n} \qquad (4.6)$$

where f_j, $(j=1,2, \ldots ,n)$, is the relative number of times file j is processed within the subsystem run cycle. The formula for z could then be used to calculate the total tape I/O time for the entire subsystem over one cycle time.

Another factor which contributes to the number of blocks processed per file is the number of reread/rewrite due to I/O error recovery attempts on the tape devices. This is not a straightforward factor to incorporate into the objective function because of its statistical nature. Furthermore, the sequence of motions which the tape goes through to retry an error block is device-dependent, (i.e. dependent on the device model and the manufacturer). In order to determine its effect, one would need to know the I/O error statistics per volume over a period of time, the user/system specified error threshold, and the operational characteristics of the tape devices. For ordinary block sizes, however, this factor can be considered negligible (for well kept tapes and read/write heads).

4.2.1 Selection of Objective

The question has been posed in section 4.1.1 as to whether the total number of IBG's processed by a subsystem is indeed the most suitable criterion for estimating incremental changes in system throughput and cost-performance ratio. To answer this question effectively, one has to first of all deal with these two notions separately.

4.2.1.1 Cost Criteria

The meaning of cost is by no means self-evident. It could refer to the dollars-and-cents cost of the total magnetic tape acquisitions of an installation. It could be in terms of the amount of core memory that can be removed from one section (partition, program) and expended for tape buffering in another. Or again it could mean the quantum jump in operating personnel cost in adding another work-shift in the computer center. Some of these cost functions might indeed involve mutually opposing trade-offs.

If the cost of tape acquisitions is being considered, then the optimization of the total number of IBG's also serves to minimize the total tape length requirement.

4.2.1.2 Throughput Criteria

The conventional meaning of throughput is more easily defined, namely the amount of useful work (the number of jobs) that is processed by the total computer system in a

given amount of time. This definition is admittedly imprecise on several points, but nonetheless self-consistent as a conceptual definition.

If our primary objective is a high throughput rate, then the amount of positive correlation between our original objective function and the system throughput time would depend largely on our frame of reference, namely the complexity of the hardware/software configuration and application program. In our basic frame of reference, the correlation is expected to be high. For instance, if a 20% reduction of total tape I/O time is estimated from the optimization in a single-stream batch processing system which is 70% I/O bound, the actual reduction of throughput time can amount up to 14% (i.e. 20% of 70%).

When this system acquires more high-speed auxiliary memory devices (disks, drums) on the same selector channel, channel contention can become a significant factor to be reckoned with. If the system grows into a multiprogramming environment, the overlapping of CPU and I/O operations will introduce further secondary effects on channel contention. Further sophistications of the system, such as time-sharing, will introduce tertiary effects of overhead from job swapping, time slicing, memory paging, etc. The interaction of all these additional factors will rule out a straightforward estimation of system performance on the basis of the proposed optimization model. In such cases, in order to be more

determinate, it will serve well to employ a simulation model. However, that does not diminish the value of this optimization approach.

4.3 Formulation of the Memory Constraints Matrix

The memory constraints matrix is formulated in light of system restrictions and memory availability. If in a rare case an individual block size (R_jX_j) obtained in the solution exceeds the channel buffer capacity, (e.g. 32 KB for the IBM/360), the associated channel capacity constraint ($R_jX_j \leq$ max. channel buffer capacity) would have to be introduced. The solution is then recomputed so that the $\max_j \{X_j\}$ lies within the upper bound imposed by the channel buffer capacity.

A more systematic approach is of course to include all the extra constraints ($R_jX_j \leq$ max. channel buffer capacity, $j=1,2, \ldots ,n$). However, the introduction of considerably more constraints creates an unnecessarily unwieldy model.[10]

To evaluate the incremental differences among various alternatives in the optimization of blocking factors, one can vary the decision parameters a_{ij} and M_i. This information can serve as an additional analytic tool in establishing memory partition sizes and job schedules.

---

[10] For instance, in a subsystem of 10 programs and 20 tape files, the original matrix of 30 constraints would be expanded to 50 constraints.

4.3.1 Effects of Manipulating a_{ij}

One can expect a certain incremental gain in the throughput of overlapped operations of CPU and I/O by increasing the number of alternate I/O buffers per tape file.[11] This increase in a_{ij} is normally effected at the expense of reducing the individual buffer sizes in a constrained memory situation. Thus this gain can be offset by the loss resulting from the increase in I/O time and overhead. This trade-off should be carefully weighed, especially when one is dealing with small blocking factors, whose incremental effects on I/O time and tape length requirement are much more pronounced.

4.3.2 Effects of Varying M_i

The incremental effect of varying the total available amount of buffer memory can be used to estimate the cost-effectiveness of expanding a memory partition, or of rescheduling a certain job into a larger partition to reduce its run time. This parameter is of major import in the formulation of the mathematical model, since many aspects of system performance differentials can be related to variations in M_i.

[11] a_{ij} is subject to a maximum number that can be handled by the I/O control system software. It is normally assigned a value of 1 or 2.

4.4 Important Considerations in System Analysis

To design an optimal system that will retain its level of performance in the face of future growth and modifications to the hardware, software or job scheduling, the analyst must necessarily make compromises in setting up a mathematical model to represent the real situation. This presupposes that information about such plans is available. However, if the system redesign can be done in an economical way, the analyst may decide to make less allowance for future changes in order to strive for better cost-effectiveness immediately. Such a 'tight' system would be subject to more frequent redesigns.

Although the primary goal of this project is to optimize tape file systems, a system analysis would be incomplete and unsound if tape operations are scrutinized in isolation without considering how they interact with other parts of the system. Such questions as the degree of CPU and I/O overlap, channel overlap, and channel contention are equally as important as the mathematical model, although they may have to be solved by a stochastic simulation model without the power and elegance of a mathematical solution.

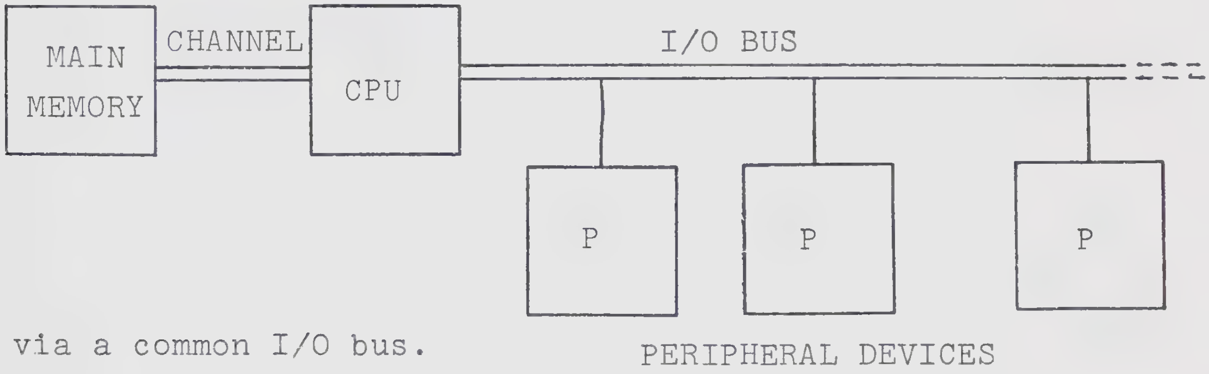
By itself, the mathematical model (memory constraints matrix) only describes the contention for main memory among the various tape files used in a program. Hence, it must be used in conjunction with the other related factors, which are summarized below:

4.4.1 Essential Information

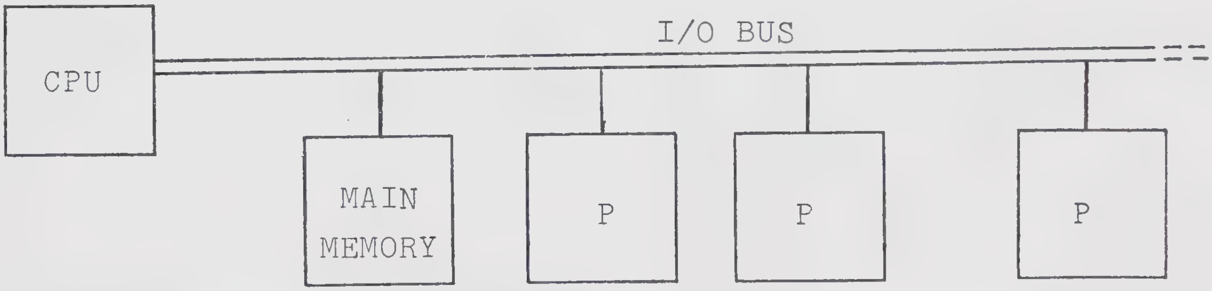
The following categories of information are considered essential input for a sound system analysis.

4.4.1.1 Hardware Configuration

In the system architecture, the analyst should ascertain whether the CPU's access to the main memory is via a dedicated channel



or via a common I/O bus.



The former arrangement is by far more common and involves less channel interference. In the latter case, which is not uncommon in small systems, memory access by the CPU may be handicapped by the malfunction of any I/O unit on the asynchronous I/O bus, or by an inordinate amount of channel contention among the peripheral units.

In addition, the number of peripheral devices

attached to each selector channel should be taken into consideration, although it is difficult to quantify this factor in our model. A system with two selector channels should have in general a lower rate of channel interference. A general intuitive assumption is that since large tape blocks monopolize the channel for longer but fewer bursts of transmission, they tend to create greater channel contention.

4.4.1.2 Software Support

The type of operating system used has a direct bearing on core memory allocation considerations, such as partition sizes (fixed or variable) and their basic unit of boundary increments. The amount of main memory available for user programs in a partition as well as the program sizes (excluding the I/O buffers) must be determined. If a program is divided into overlaying segments, the available buffer memory may be limited by the size of the largest segment.

4.4.1.3 Master System Flowchart

A flowchart (or an equivalent description) of each application subsystem showing the interrelations of individual program runs and their associated files is indispensable. In addition to providing an effective means to highlight the existence of bottlenecks in a system, its modular representation also serves as an aid to analyzing different alternatives of job scheduling.

4.4.1.4 System Audit Data

The statistical breakdown of audit data on system usage indicates the amount of usage of the CPU and I/O devices per program and per application. Percentage utilization of these resources and the throughput rate can be calculated from these cumulative data. The amount of I/O overhead and channel interference can also be gauged indirectly from these statistics, (e.g. the number of I/O waits during a particular time interval).

4.4.1.5 File Characteristics

Files are normally organized in ways that are most efficient for their storage and retrieval. Magnetic tape files are generally formatted with fixed or variable length records that can be blocked. The choice between using fixed or variable size records is usually made on the basis of the file activity and the record size variability. As such, this design parameter is considered independently of the blocking criterion, and hence does not enter into the mathematical formulation of this optimization problem.

In the case of a system redesign, the existing information on blocking factors (X_j), record sizes (R_j), block sizes (R_jX_j), normal volume of files (F_j), and the number of alternate I/O buffers (a_{ij}) per file per program will serve as a basis for the a priori estimation of performance gains and the a posteriori evaluation of cost-effectiveness of alternative system designs.

4.4.1.6 Sort/Merge Utilities

These utility programs very often impose an implicit constraint on the maximum practical block size, above which the system performance becomes significantly degraded. (E.g. in the case of disk sorts, the sorting efficiency is significantly reduced if the block size exceeds one track on the disk, which is 3625 bytes on the IBM/2311 disk or 7294 bytes on the IBM/2314 disk.)

4.4.2 Desirable Information

The following categories of input information are considered highly desirable but not essential for the system analysis. Some of them may not be easily obtainable.

4.4.2.1 Plans for Future Growth

As far as possible, the analyst should take cognizance of long-range and short-range future plans for changes in the hardware configuration, software support, application programs or job scheduling. These plans may range from specific short-term modifications to ill-defined and sweeping long-range objectives.

Take for instance an immediate plan for the acquisition of an extra memory module. This would leave ample room for a major revision of the memory constraints of each subsystem. A ramification or goal of this decision may be the rescheduling of entire application subsystems, facilitated

by the major improvement in system performance. Other changes might be more localized, such as the rescheduling of a particular job into a larger memory partition, or more sweeping, such as a major systems conversion.

Obviously, no hard and fast rules can be laid down on how much significance should be attached to certain anticipated changes with respect to the formulation of a mathematical model of the system. What can and should be stressed in this regard is that any system analysis must be sufficiently circumspective of unusual conditions or unanticipated changes that could make the system inoperative. These considerations may be the least tangible or amenable to mathematical formulation, but in more profound ways determine the actual payoff and indeed the operational lifespan of a system design.

4.4.2.2 Peak Loads

Most systems are designed with enough capacity to handle estimated periodic peak loads. The drawback of this approach is the less-than-full utilization of available resources during non-peak periods. Other systems are designed to handle the average volume most efficiently, with the excess load at peak periods handled inefficiently or siphoned off to be processed by another system.

The volume of transactions is such a case in point. During the cycle time of a subsystem, a transaction file or a master file may grow or subside in size. The pattern of

variation may have periodic characteristics over a daily, weekly, monthly, or a yearly cycle. For instance, a transaction file may grow steadily from 5,000 records in January to 9,000 records at year-end. At mid-year it may have an average size of 7,000 records. If this file size F_j is estimated at the mid-year figure of 7,000 records, and its optimal blocking factor is calculated based on this, then its I/O efficiency is expected to be greater than what it should be during the first half of the year (at the expense of other files in the subsystem), and less than optimal during the latter half of the year. This phenomenon is by no means unique to file design. But it does represent the kind of compromise that will have to be made in many instances in order to achieve an overall optimum. Other important measures of peak loads include the run time per program, per subsystem, or per work-shift.

4.4.2.3 System Bottlenecks

Bottlenecks in a system may be classified into two general categories, namely in performance or in scheduling. Performance bottlenecks are considered more localized. They can occur in a program loop or segment that consumes the bulk of the processing time or I/O resources of a program or subsystem during its execution, such as a sort program. Scheduling bottlenecks, on the other hand, tend to be more generalized. They can be a tight deadline which a job or a whole application

is expected to meet, or the introduction of an additional program into the production job stream of a tight work-shift. These considerations definitely influence the formulation of the mathematical model and the a posteriori adaptation of its solution.

In a paper on multiprogramming systems performance measurement and analysis, Cantrell and Ellison [6] presented some poignant observations on the crucial effects of bottlenecks on the performance of a system. They can be illustrated by the following examples:

A certain frequently used program might involve four tape files running in a system with four tape devices. The input and output master files both have grown to occupy a little over one full reel. The run time of this job would increase by a significant amount due to the extra tape rewind and set-up time. This problem may be avoided by introducing a tape length constraint,

$$\frac{F_j}{X_j} \times (G + D \times B) \leq L_j \tag{4.7}$$

where G, D and B denote the same parameters as in equation 3.3, and L denotes the length of a full reel (less the length required for label information). In a similar manner, one could impose an I/O time constraint,

$$\frac{F_j}{X_j} \times (S + C \times B) \leq T_j \qquad (4.8)$$

where S, C and B denote the same parameters as in equation 3.2, and T_j denotes the total I/O time which must not be exceeded by file j.

In trying to incorporate these two new dimensions of constraints, namely length and time into the original memory constraints matrix, a mixed type constraints matrix is thus created:

$$\begin{array}{ccccccccccc}
a_{11}R_1X_1 & + \ldots + & a_{1n}R_nX_n & + & 0 & + & 0 & \leq M_1 \\
a_{m1}R_1X_1 & + \ldots + & a_{mn}R_nX_n & + & 0 & + & 0 & \leq M_m \\
0 & + \ldots + & 0 & + & \frac{F_j}{X_j}(G+D\times B) & + & 0 & \leq L_j \\
0 & + \ldots + & 0 & + & 0 & + & \frac{F_k}{X_k}(S+C\times B) & \leq T_k
\end{array} \qquad (4.9)$$

Besides introducing new variables as well as extra constraints into the model, this extended matrix combines linear and non-linear constraints. This mixed type matrix requires more elaborate methods of treatment. Of course, a transformation of variables can be made such that a constraint becomes linear.

$$\frac{X_j}{F_j(G+D\times B)} \geq \frac{1}{L_j} \qquad (4.10)$$

But this does not alleviate the problem of solving for a unique X_j for a particular file occuring in different types of constraints. Furthermore, a composite objective function would need to be defined in accord with the composite constraints matrix. The different components of any such overall measurement of system performance would have to be weighted empirically. In this general area, E.A. Silver et al [29] have proposed a weighting algorithm to aid the systematic allocation of system resources.

In our present context, the above-mentioned refinements would create an excessively elaborate and unwieldy mathematical model, both from the points of view of system analysis and finding available algorithms for its solution. The payoff for this elaboration of the problem is not immediately apparent. Indeed, the complexity may deter the management or analysts from undertaking this project.

However, one can take a crudely empirical approach to achieve the desired results. In substance, one can artificially bias the parameters of the affected files in the model. The best predicted solution is then adopted. It should be re-emphasized here that the overriding objective is to achieve useful results with the minimum necessary effort and a sound methodology. With due regard to its effects on the physical system, a mathematical model can be distorted in order to simplify its algorithm for solution.

4.4.2.4 File Activities

Application files normally vary in size and pattern of activity, often in regular cycles. The relative amount and manner by which the file sizes vary affect the coefficients of the objective function directly. Hence, any meaningful estimation of a highly variable file size over a period of time would need to take into consideration the pattern of variation.

Does the file size fluctuate with a uniform or normal distribution, or skewed towards a short and rapid growth followed by a long and slow decline, or vice versa ? Does the mean size coincide with the median size, and which one is more representative ? What is the variance of a file size over a period of time, say, a year ? What is the pattern of activity in the master files ? Are the transactions uniformly distributed, or clustered around a small percentage of high activity accounts ? In the latter case, the majority of the master records can be read/written 'on the fly', that is, without necessarily waiting for the tape motion to come to a complete stop in the IBG. By itself, this consideration does not directly affect the mathematical solution for the optimal blocking factors. However, the file size coefficient of a low activity file could be artificially reduced to take advantage of this fact, (under the assumption that a relative increase in I/O time resulting from a decrease in the blocking factor for a low activity file would be more than compensated

by the increase of overall I/O efficiency of the subsystem.)

4.4.2.5 Statistical I/O Error Rates

Statistical information on I/O error rates is normally collected per file volume and tape device. This information can be used to gauge the relative significance of the number of error block retries for a particular tape file.

Theoretically, the larger the block, the higher is the probability of occurrence of an I/O error in that block. However, the total number of blocks in the file is reduced. In any case, a reasonably well maintained tape installation is not likely to encounter a situation where the number of I/O error blocks per file becomes significant enough to affect the formulation of the objective function with error terms.

4.4.2.6 Selection Criteria for Blocking Factors

In order to make a meaningful comparison of the performance improvements that can be expected from a mathematical solution, the analyst would need to know the criteria by which the blocking factors were chosen in the past. Was the approach purely ad hoc ? As is often likely, each subsystem was probably given the blocking factors that represent the best guess of a systems analyst. In a situation with system constraints, this is seldom the optimal arrangement, and a sensitivity analysis of the actual and the optimal arrangements would indicate the amount of deviation in system performance.

CHAPTER V

EXTENSION OF PROBLEM

5.1 Direct Access Storage Devices

The foregoing chapters have identified and treated a problem in optimization of computer system resources in a relatively straightforward and predictable frame of reference. The environment considered was a non-overlapped, single batch processing with tape files. In reality, few data processing systems fit such a simplified description. With the addition of direct access files and the introduction of more powerful operating systems, the question arises as to whether the effects of varying the blocking factors on disk and tape files are mutually unrelated and additive, or are related in some non-linear fashion. The remainder of this chapter attempts to address this problem.

5.2 Disk File Organization

Disk files are generally organized in one of the four methods: sequential, partitioned, indexed sequential, or direct.

5.2.1 Sequential Organization

In a sequential file, records are organized solely on the basis of their successive physical locations in the file. The records are generally, but not necessarily, in

sequence according to their control keys as well as in physical sequence. The records are usually read or updated in the same order in which they appear.

5.2.2 Partitioned Organization

A partitioned file is one that is divided into several members. Each member has a unique name. Members may be called by name for processing. The records within the members are organized sequentially and are retrieved or stored successively according to physical sequence.

5.2.3 Indexed Sequential Organization

An indexed sequential file is similar to a sequential file in that rapid sequential processing is possible. Indexed sequential organization, however, by reference to indices associated with the file, makes it also possible to quickly locate individual records for random processing.

5.2.4 Direct Organization

A file organized in a direct manner is characterized by some predictable relationship between the key of a record and the address of that record on the direct access storage device (DASD). This relationship is normally established by a randomizing algorithm that converts a record key into a unique DASD address.

5.3 Disk Storage Organization

A disk device essentially consists of a number of recording tracks per disk surface. Each set of tracks that can be accessed by the read/write heads by the same positioning of the movable access arms forms a cylinder.

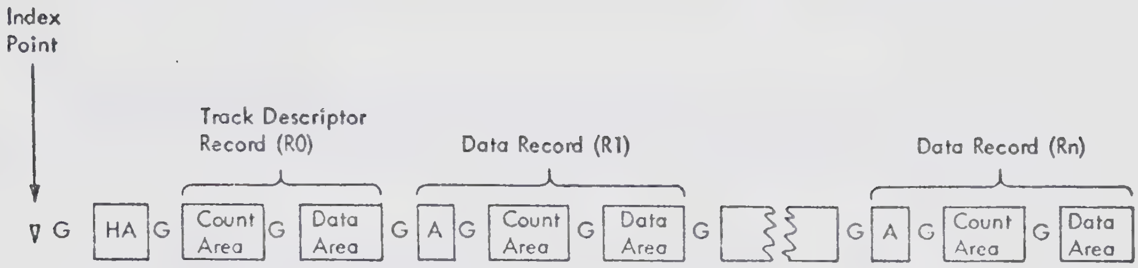
5.3.1 Track Formats

Information is recorded on all devices in a format which is prescribed by the control unit and which is identical for all devices. Each track contains certain 'non-data' information as well as data information. The non-data information generally consists of track alignment index points, physical gaps, block addresses and record address. The data records may be formatted with external keys (count-key-data) or without external keys (count-data). Figure 5.1 illustrates the track formats for one particular manufacturer.

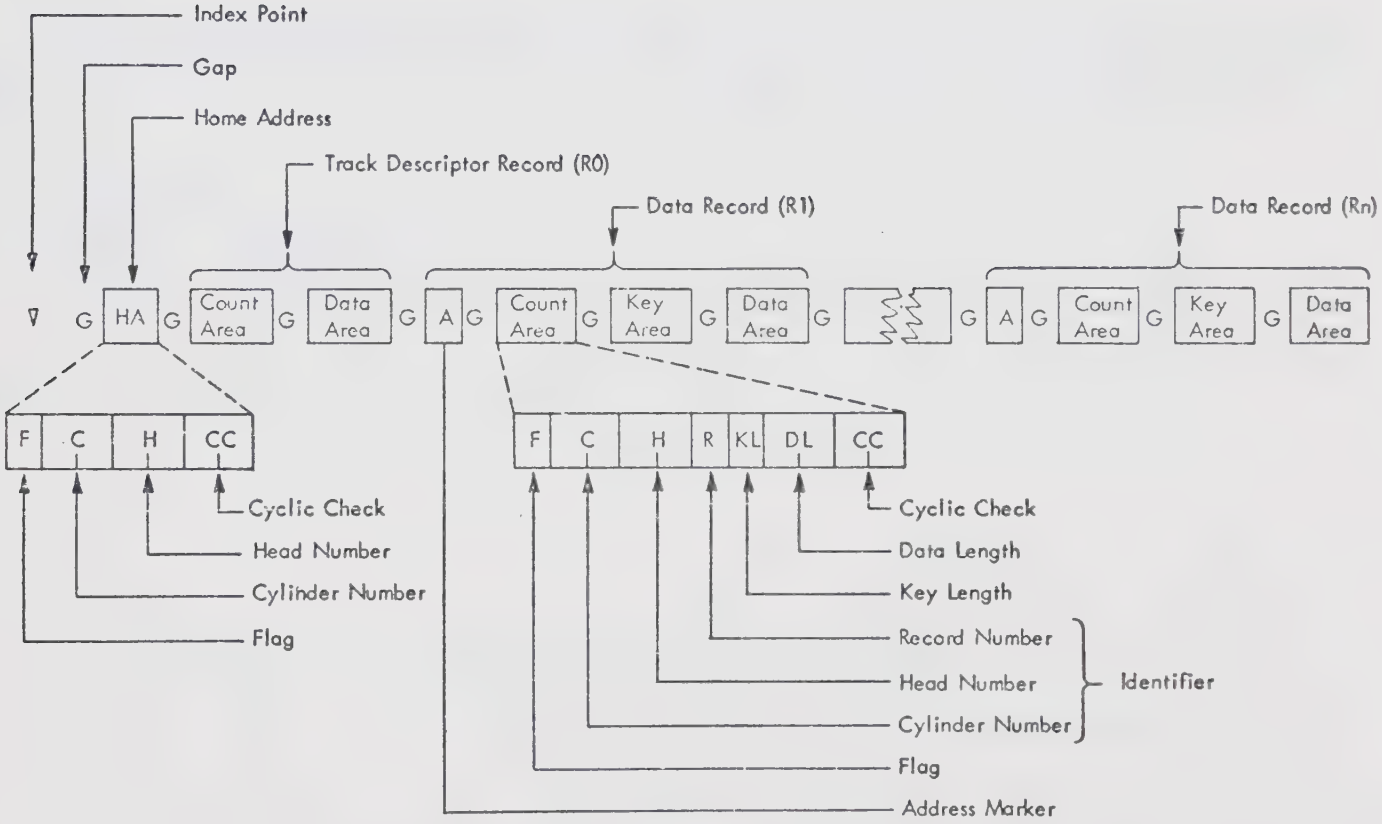
5.3.2 Record Formats

Logical records may be organized in one of five formats (as illustrated in figure 5.2):

- Fixed unblocked
- Fixed blocked
- Variable unblocked
- Variable blocked
- Undefined



A. Count-data format



B. Count-key-data format

Figure 5.1 DASD track formats. (IBM)

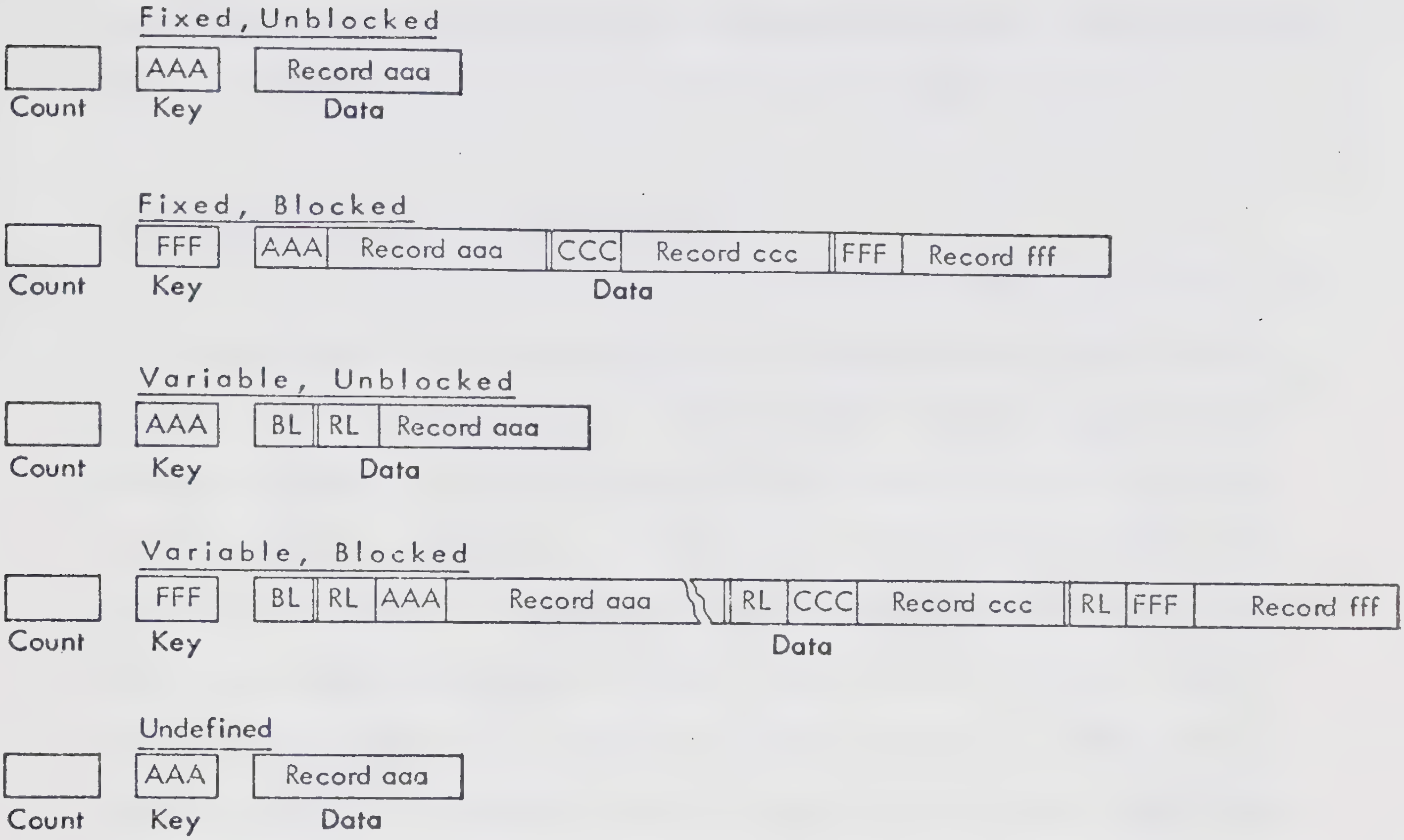


Figure 5.2 DASD record formats.

5.4 Effects of Blocking on DASD File Systems

As one can observe in figure 5.1, physical gaps analogous to the inter-block gaps on magnetic tape files also exist between records on DASD files. It would thus seem natural to apply the same methods of treatment as for tape files. In fact, however, the operational characteristics of DASD are significantly different from those of tape devices. The two basic device characteristics which distinguish DASD

from tape devices in blocking considerations are their constant speed of rotation and their fixed track length.

5.4.1 Effects on File Compaction

The primary reason for blocking records in DASD files is to pack the file storage more efficiently and hence reduce storage space requirements. With blocked records, gaps exist between blocks of records rather than between each individual logical record. However, it should be noted from figure 5.3 that the file space requirement does not decrease monotonically with increasing blocking factors. This is due to the fact that a block cannot be spread over two tracks.[12] Thus, for certain blocking factors, the last block on a track overflows one track and the entire block is pushed onto another track. This results in some wastage per track, and it becomes more significant as blocks become larger. These relative increases in the amount of storage space wastage are represented by the local maxima in figure 5.3. The same type of characteristic curve applies to any track-formatted DASD (i.e. disk or drum) which does not allow a block to be split over two tracks.

[12] Some DASD are equipped with a feature for splitting a block over two tracks. In such a case, the characteristic curve will resemble those of tape devices, i.e., there are no local maxima due to track boundaries. However, they still exist due to cylinder boundaries, though fewer than before, intuitively.

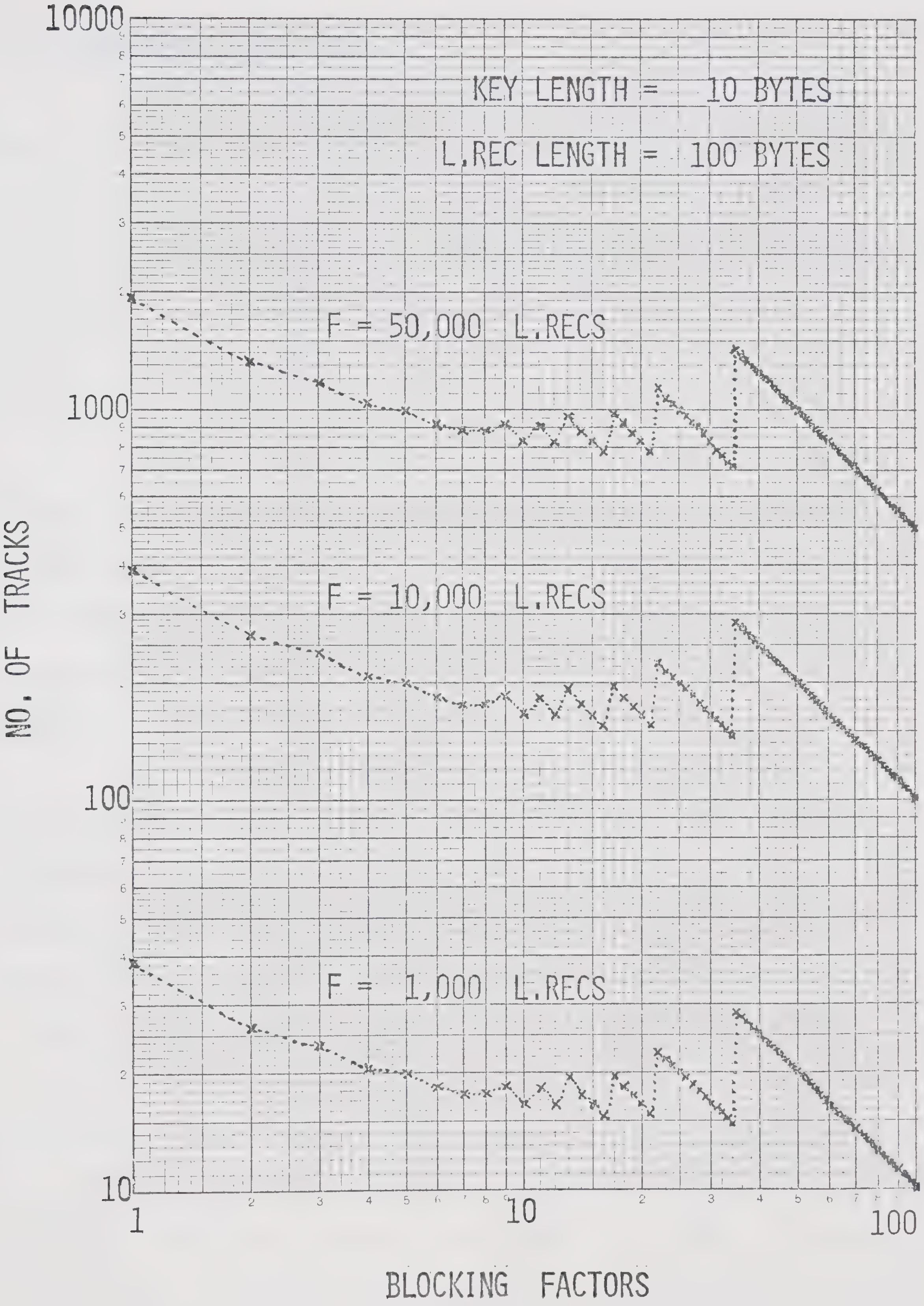


FIG. 5.3 NO.OF TRACKS REQUIRED PER FILE AS FUNCTION OF B.F. (IBM/2314 DISK DEVICE, WITH KEYS)

5.4.2 Effects on I/O Time

Another reason for blocking is that it may save time. If records are processed consecutively, there is only one rotational delay before reading or writing a block of records. However, if records are not processed consecutively, blocking may be a disadvantage, since it takes longer to transfer the entire block rather than the single record to be processed.

A more compelling reason for blocking records on disk devices with movable heads is that the file compaction reduces the total number of cylinders required by the file. If the DASD file is assigned a group of contiguous cylinders, this reduction results in a smaller average number of cylinders traversed by the read/write heads.

In a directly organized disk file, where the random input transactions require a short response time, the trade-off between an increase in block transfer time and a reduction in the average seek time depends on several related factors, namely the distribution of high activity records in the file and the physical scatter of the file into non-contiguous cylinders. Here one must distinguish between disk devices with movable heads and drum devices (or fixed-head-per-track disk devices). In the former, the block transfer time is less significant than the average seek time, of a ratio of 1 to 5. In the latter, however, the seek time is negligible (at electronic speeds).

5.5 Isolation of Problem

From the foregoing discussion, it should become obvious that not all forms of processing or methods of DASD file organization lend themselves to the straightforward kind of mathematical modelling that was applied to tape file systems. For instance, in a directly organized drum (or fixed head disk) file that is processed randomly, the optimization problem hinges on a space-time trade-off involving a qualitative judgement which is beyond the objective of this optimization problem, i.e. to minimize I/O time or storage space. This is not to say that the method of treatment cannot be further elaborated to deal with this particular situation. But for the purposes of this dissertation, this case can be isolated and removed from consideration.

We shall therefore only consider DASD files on disk devices with movable heads. The working assumption made here is that the reduction of the average seek time more than compensates for the increase in block transfer time when records are blocked optimally. This approach would thus apply also to random processing of directly organized files.

For reasons of functional modularity and analytical simplicity, the disk file subsystem will be considered separately from tape file subsystems. Their separate mathematical models may lead to sub-optimal solutions. But the functional disparities between the two types of subsystems often necessitate different requirements for modification and reorganization.

5.6 Methodology of Treatment

5.6.1 Formulation of Objective Function

The objective function for disk file subsystems is formulated differently from that of tape file subsystems. The primary objective here is to minimize the total number of tracks required by a disk file. Because of the fixed track length, the minimization of the total number of disk record blocks does not necessarily imply a minimization of the total number of tracks required. This fact is illustrated by the existence of local maxima on the graph of figure 5.3.

The total number of tracks to be minimized for n disk files is given by

$$z = \sum_{j=1}^{n} f_j \left(\frac{F_j}{X_j \; Y_j}\right) \qquad (5.1)$$

$$Y_j = \left(\frac{U - V_j}{W_j}\right)^* + 1 \qquad (5.2)$$

$$V_j = K_j + B_j + C \qquad (5.3)$$

$$W_j = P \; (K_j + B_j)^* + C + Q \qquad (5.4)$$

---

* The fractional parts are truncated after the operation.

where f_j, F_j, X_j have the same meaning as in equation 4.6,

z = Total no. of tracks required for n disk files

Y_j = No. of blocks per track for disk file j

U = Track capacity in no. of characters

V_j = No. of characters in last block on track

W_j = No. of characters per block (except last block)

K_j = Key length in no. of characters for file j

B_j = No. of characters per block for file j

C = Device constant ($C=0$ if $K_j=0$, $C>0$ if $K_j\neq 0$)

P = Device constant (allows for positional deviations)

Q = Device constant for non-data information in no. of characters

5.6.2 Formulation of Memory Constraints Matrix

The memory constraints matrix for a DASD file subsystem has the same form as in formula 4.5. But although it has the same form, more care should be taken in setting it up in conjunction with the performance criteria which are not quantified into the mathematical model, such as channel configuration, DASD file organization and access methods.

5.6.3 Discrete Optimization Methods

Because of the nature of the function as illustrated by the local maxima in Fig. 5.3, the optimization problem is best solved by using integer programming methods. [90, 91, 93] This class of algorithms is beyond the scope of this thesis, and hence will not be discussed beyond listing some key references.

A possible alternative approach is to approximate the discrete function with a polynomial and then proceed to optimize the continuous function.

CHAPTER VI

IMPLEMENTATION

6.1 Research Recommendations and Managerial Action

The subject of operations research implementation has been treated in depth by Huysmans [52]. In his book, Huysmans stresses the importance of the involvement of management at all levels in order to facilitate a smooth and successful implementation of the OR recommendations. The failure to secure top management agreement on basic objectives and scope of the project at the initial planning stage may jeopardize the acceptance of the recommendations at the implementation phase. The possible behavioral reaction of the manager to the research recommendations as a function of managerial understanding can be classified in one of four categories:

- rational rejection
- resistance
- acceptance
- implementation

6.1.1 Rational Rejection

The manager understands the research recommendations, but rejects them on rational grounds. The research may be inadequate, irrelevant, or not superior to present practice (transition cost considered). Irrelevance of the research

often takes the form of an improper consideration of a subsystem of the total system. That is, the wrong constraints have been singled out for optimization.

Rational rejection is related to the problem of project selection, that is, the choice of a subsystem to be considered in the research. It should be realized that the evaluation of potential implementation problems should be an important determinant of project selection.

6.1.2 Resistance

The manager rejects the research recommendations, and his way of thinking about the research problem deviates considerably from that of the analyst's. Rejection in this case may very well be caused by a lack of managerial understanding. If the analysis is adequate, relevant, and superior to present management practice, there exists an implementation problem. The manager may fail to recognize the advantages of the research recommendation for various reasons. He may not have paid sufficient attention to the research proposal; he may incorrectly evaluate the proposal's return, cost, or transition cost; his personal goals may conflict with and take precedence over organizational objectives; or he may have insufficiently resolved the inconsistencies in his perception of organizational objectives.

6.1.3 Acceptance

The manager adopts the research proposal, but does not understand it. An implementation problem exists to the extent that instability is introduced by mere acceptance of the research recommendation.

6.1.4 Implementation

The manager understands the research proposal and adopts it. It is useful to divide this category into two sub-categories:

- sustained implementation
- autonomous implementation

6.1.4.1 Sustained Implementation

Managerial understanding of the research recommendations is integral, that is, the manager has good overall understanding of the critical factors underlying the recommendations, but this understanding is acquired at the cost of continued involvement of the analyst. The manager realizes that changes in environmental parameters may require changes in his actions. He is able to identify the change points, but proper reaction to the changes is only assured if the analyst stands ready to provide his continuous support.

6.1.4.2 Autonomous Implementation

Managerial understanding of the research is explicit and complete. No continued support of the analyst is required. Adoption occurs because the analyst's proposal is consistent with the manager's perspective and solves a problem for him. The source of the proposal is of minor importance in this case.

6.2 System Performance Monitoring

In the initial testing phase of a designed subsystem, benchmarks are employed to simulate and measure its performance under actual load conditions. Once an optimized subsystem has been implemented on a regular production schedule, its performance is monitored and evaluated regularly. At any stage of the system life cycle, the feed back from one phase may necessitate modification of preceding phases. The monitored performance of a subsystem thus ultimately determines the validity and reliability of this OR approach.

CHAPTER VII

CONCLUDING REMARKS

7.1 Economic Payoffs and Intangible Benefits

The foregoing chapters have described an operations research approach to solve a problem of optimal resource allocation in tape and disk file systems. Analysts who are convinced of the immediate payoffs in increase of throughput and reduction of file storage requirement still face the task of convincing the management of the advantages of this mathematical optimization approach. This approach to problem solving will have to be sold to the management in terms of management's frame of reference -- namely, system objectives, project development costs and economic payoffs.

In this light, the file compaction advantage may be translated into the following management considerations:

- reduce tape and disk acquisition costs
- improve job scheduling

Similarly, the reduction of I/O time and overhead may be translated into the following management terms:

- increase throughput (or reduce computer time cost)
- reduce bottlenecks in job scheduling

What may be equally valid and perhaps have more pervasive impact are the intangible benefits:

- greater systems awareness
- improved resource allocation
- better management control

7.2 Applicability in Future Technology

The costs of memories at all levels are continually decreasing as technology advances. Already, it is reaching a point where the price/performance ratio at each level is becoming competitive with that of the next lower level. Some installations are now using large scale main memory for data which used to reside on DASD several years ago. Disk and drum devices have also evolved to the stage of sophistication where the price/performance ratio makes them competitive with the old magnetic tape technology, which itself is reaching new heights of speed and economy.

However, with full recognition of this rapid technological obsolescence that characterizes the computer industry, the upgrading and diversification of hierarchical memories will not invalidate this mathematical optimization approach to computer systems resource allocation. In fact, the present trend of increasing complexity in computer systems can only lead to a greater reliance on such analytical techniques.

APPENDIX A

Local Optimization by Lagrange Multiplier Method

APPENDIX A

LOCAL OPTIMIZATION BY LAGRANGE MULTIPLIER METHOD

Let us consider the following example of a Master Update program:

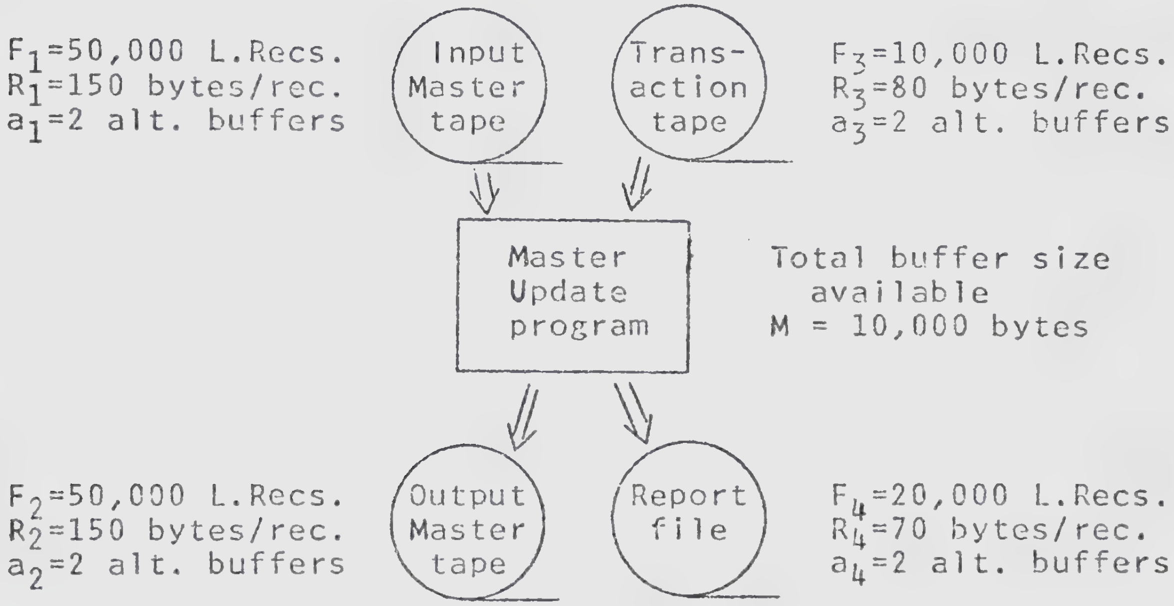


Fig. A-1 An application program using four tape files.

The objective function (with h.c.f. factored out) is

$$z = \frac{5}{X_1} + \frac{5}{X_2} + \frac{1}{X_3} + \frac{2}{X_4} \qquad \text{(i)}$$

The memory constraint equation is

$$2 \times (150 \times X_1 + 150 \times X_2 + 80 \times X_3 + 70 \times X_4) = 10{,}000 \qquad \text{(ii)}$$

Taking partial derivatives of (i) and (ii) with respect to X_j, ($j=1,2,3,4$), we obtain the simultaneous equations

$$-\frac{5}{X_1^2} + 300\lambda = 0 \qquad \text{(iii)}$$

$$-\frac{5}{X_2^2} + 300\lambda = 0$$

$$-\frac{1}{X_3^2} + 160\lambda = 0$$

$$-\frac{2}{X_4^2} + 140\lambda = 0$$

Solving the equations (ii) and (iii) simultaneously, we obtain the mathematically optimal combination of blocking factors

$$X_1=12.1, \quad X_2=12.1, \quad X_3=7.4, \quad X_4=11.2$$

which we can round off heuristically to give the physically optimal combination (occupying exactly 10,000 bytes)

$$X_1=X_2=12, \quad X_3=7, \quad X_4=12$$

From this we can determine the total *I/O* time for the four files

$$T = 158.58 + 158.58 + 24.79 + 36.71 \text{ seconds}$$

$$= 378.86 \text{ seconds}^*$$

---

* Example using the formula for IBM/2401 mod 2 (800 bpi) 9-track tape device.

Sensitivity Analysis

If the BF's were arbitrarily assigned, say,

$$X_1' = X_2' = X_3' = X_4' = 11,$$

(with a slack of 100 bytes to spare), the total I/O time would be

$$T' = 161.61 + 161.61 + 20.63 + 37.92 \text{ seconds}$$

$$= 381.78 \text{ seconds}$$

which differs from the optimal T by

$$\Delta' = (T' - T) / T = 0.8 \%$$

If on the other hand the total core memory available for tape buffers were equally divided among the four files, thus assigning $X_1'' = X_2'' = 8$, $X_3'' = 15$, $X_4'' = 20$, (occupying exactly 10,000 bytes), we would obtain

$$T'' = 175.25 + 175.25 + 18.69 + 31.38 \text{ seconds}$$

$$= 400.57 \text{ seconds}$$

which differs significantly from the optimal T by

$$\Delta'' = (T'' - T) / T = 5.7 \%$$

The different degrees of file compaction for the above cases can also be calculated using a similar formula. It should be noted in passing that the percentage differences remain the same for the same relative difference in file sizes.

```
C     PROGRAM TO SOLVE FOR OPTIMAL TAPE BLOCKING FACTORS
C     BY LAGRANGE MULTIPLIER METHOD.
C
C     FORMAT FOR INPUT DATA:
C     CARD 1
C           COLS 1-3     NO. OF OBJECTIVE FUNCTIONS (INTEGER)
C     CARD 2     ONE CONSTRAINT PER OBJECTIVE FUNCTION
C           COLS  1- 3   NO. OF TAPE FILES (INTEGER)
C           COLS  4-13   MEMORY AVAILABLE FOR TAPE BUFFERS
C     CARD 3             RECORD SIZES
C           10 COLS PER FIELD, 8 FIELDS PER CARD (DECIMAL)
C     CARD 4             FILE SIZES
C           10 COLS PER FIELD, 8 FIELDS PER CARD (DECIMAL)
C     SUCCEEDING CARDS REPEAT FORMAT FOR CARDS 2 TO 4.
C
      DIMENSION X(9), R(9), F(9), T(9), CON(9)
      READ(5,101) M
      WRITE(6,100)
      DO 200 J=1,M
      READ(5,101) N, C
      READ(5,102) (R(I),I=1,N)
      READ(5,102) (F(I),I=1,N)
C     BUFFER MEMORY/2  FOR 2 ALTERNATE BUFFERS PER TAPE FILE
      CK=C/2.
      S=0.
      DO 130 I=1,N
      T(I)=SQRT(R(I)*F(I))
130   S=T(I)+S
      W=S/CK
      DO 140 I=1,N
140   CON(I)=R(I)*2.
      WRITE(6,103) (CON(I),I=1,N),C
      WRITE(6,104)
      DO 150 I=1,N
      X(I)=SQRT(F(I))/(SQRT(R(I))*W)
150   WRITE(6,105) I,X(I),I,R(I),I,F(I)
      Z=0.
      DO 180 K=1,N
180   Z=Z+F(K)/X(K)
      WRITE(6,106) Z
200   CONTINUE
100   FORMAT('1')
101   FORMAT(I3,F10.0)
102   FORMAT(8F10.0)
103   FORMAT(//,'0MEMORY CONSTRAINT EQUATION :',//,16F8.0)
104   FORMAT('0OPTIMAL B.F.''S:',T21,'RECORD SIZES:',
     *T41,'FILE SIZES:',/)
105   FORMAT(' X(',I2,') =',F6.2,T21,'R(',I2,') =',F5.0,
     *T41,'F(',I2,') =',F8.0)
106   FORMAT('0MINIMUM VALUE OF OBJECTIVE FUNCTION  Z =',F8.0)
      STOP
      END
```

```
MEMORY CONSTRAINT EQUATION :

   300.    300.    160.    140.  10000.

OPTIMAL B.F.'S:    RECORD SIZES:       FILE SIZES:

X( 1) = 12.08      R( 1) = 150.        F( 1) =  50000.
X( 2) = 12.08      R( 2) = 150.        F( 2) =  50000.
X( 3) =  7.40      R( 3) =  80.        F( 3) =  10000.
X( 4) = 11.19      R( 4) =  70.        F( 4) =  20000.

MINIMUM VALUE OF OBJECTIVE FUNCTION  Z =  11415.


MEMORY CONSTRAINT EQUATION :

   300.    200.  12000.

OPTIMAL B.F.'S:    RECORD SIZES:       FILE SIZES:

X( 1) = 31.79      R( 1) = 150.        F( 1) =  50000.
X( 2) = 12.31      R( 2) = 100.        F( 2) =   5000.

MINIMUM VALUE OF OBJECTIVE FUNCTION  Z =   1979.


MEMORY CONSTRAINT EQUATION :

   200.    200.   8000.

OPTIMAL B.F.'S:    RECORD SIZES:       FILE SIZES:

X( 1) = 30.39      R( 1) = 100.        F( 1) =   5000.
X( 2) =  9.61      R( 2) = 100.        F( 2) =    500.

MINIMUM VALUE OF OBJECTIVE FUNCTION  Z =    217.
```

APPENDIX B

GLOBAL OPTIMIZATION BY GRADIENT PROJECTION METHOD

APPENDIX B

GLOBAL OPTIMIZATION BY GRADIENT PROJECTION METHOD

Let us consider the following subsystem of 3 programs and 6 tape files:

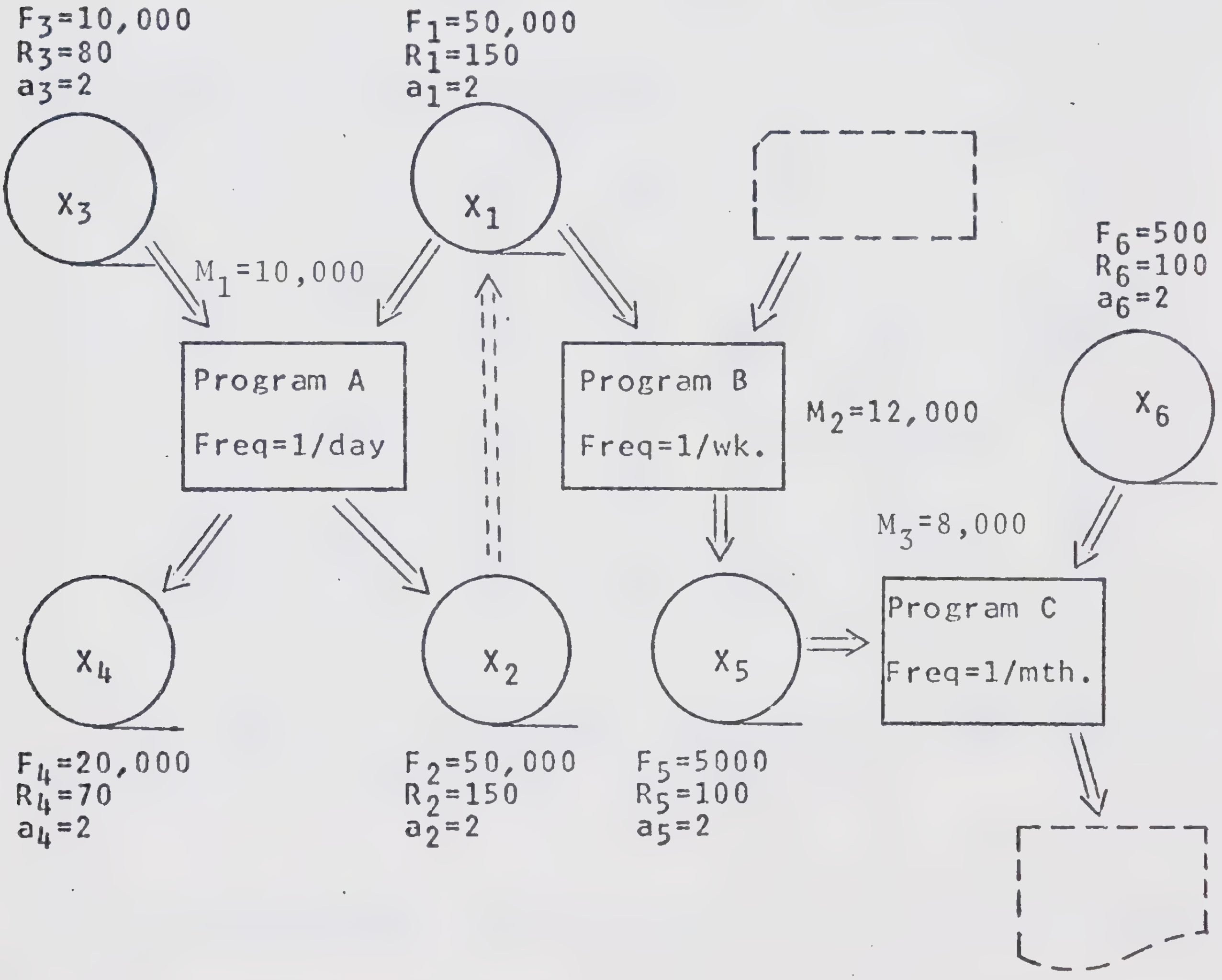


Fig. B-1 A subsystem of 3 interrelated programs and 6 tape files.

The objective function is

$$z = 24.25 \frac{(50000)}{X_1} + 24.25 \frac{(50000)}{X_2} + 22 \frac{(10000)}{X_3} + 22 \frac{(20000)}{X_4} + 5.5 \frac{(5000)}{X_5} + 1 \frac{(500)}{X_6}$$

The memory constraints matrix is

$$\begin{array}{rrrrrrcr}
X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & & M_i \\
 & & & & & & & \\
300 & 300 & 160 & 140 & 0 & 0 & \leq & 10000 \\
300 & 0 & 0 & 0 & 200 & 0 & \leq & 12000 \\
0 & 0 & 0 & 0 & 200 & 200 & \leq & 8000 \\
 & & & & & & & \\
1 & 0 & 0 & 0 & 0 & 0 & \geq & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & \geq & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & \geq & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & \geq & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & \geq & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & \geq & 0
\end{array}$$

With the use of a nonlinear program, the optimal solution for the tape blocking factors was found to be

$$X_1 = X_2 = 12.2, \quad X_3 = 7.1, \quad X_4 = 10.8, \quad X_5 = 35.2, \quad X_6 = 4.8$$

which could be rounded off heuristically to the physical optimum

$$X_1 = X_2 = 12, \quad X_3 = 7, \quad X_4 = 12, \quad X_5 = 36, \quad X_6 = 4$$

The buffer memory used, tape lengths, and *I/O* times per run of each file were computed to be

```
                                                                6.00813
IOTIME (SEC.) = 394.696  394.696  76.2827  104.55   21.489 ←
LENGTH (FEET) = 616.753  616.753  119.196  163.368  33.5793←
MEMORY USED   = 3600  3600  1120  1680  7200  800
                                                                9.38802
```

The minimum value of the objective function, i.e., the total number of tape blocks processed in one run cycle of the subsystem, (one month in this example,) is z = 271067.

Over one run cycle, the *I/O* time per file becomes

```
TOTAL                                                           6.0081
   IOTIME (SEC.) = 9571.38  9571.38  1678.22  2300.11  118.19←
```

It is immediately obvious that any significant percentage reduction in the *I/O* times for files 1 and 2 would improve the system performance much more substantially than if the same percentage reduction were effected on files 3 to 6.

In general, large tape files that are run frequently should be given large blocking factors.

Sensitivity Analysis :

If the tape blocking factors for program A were assigned arbitrarily, say, $X_1=X_2=X_3=X_4= 11$,

and those for programs B and C were subsequently assigned, say, $X_5=X_6= 20$, then we obtain the results

```
                                                                2.53483
Z = 281855
IOTIME (SEC.) = 407.853  407.853  58.2395  109.813  25.3483 ←
LENGTH (FEET) = 637.311  637.311  91.0038  171.591  39.6094 ←
MEMORY USED   = 3300  3300  1760  1540  4000  4000
                                                                3.96094
```

Over one run cycle of the subsystem, the I/O time per file becomes

```
                                                           2.5348
TOTAL
  IOTIME (SEC.) = 9890.42   9890.42   1281.27   2415.89   139.415
```

Percent deviations from the optimum can then be computed from the formula

$$\Delta' = \frac{\sum_j f_j \, (T_j' - T_{j(opt)})}{\sum_j f_j \, T_{j(opt)}} \times 100 \ \%$$

where f_j and T_j denote the same as defined earlier.

Thus,

$\Delta' =$

```
                                                              -57.8
 PERCENT TIME DIFF FROM OPTIMAL = 3.33   3.33   -23.7   5.03   18
 PERCENT LENG DIFF FROM OPTIMAL = 3.33   3.33   -23.7   5.03   18
 PERCENT TOTAL TIME DIFF FROM OPTIMAL = 1.61
 PERCENT TOTAL LENG DIFF FROM OPTIMAL = 1.39                  -57.8
```

If the buffer memory for program A were divided equally among its four tape files, and then files 5 and 6 for programs B and C were readjusted arbitrarily, assigning

$$X_1 = X_2 = 8, \quad X_3 = 15, \quad X_4 = 20, \quad X_5 = 39, \quad X_6 = 1,$$

then we obtain the results

```
                                                               19.033
 IOTIME (SEC.) = 467.057   467.057   49.8194   81.3951    21.118
 LENGTH (FEET) = 729.818   729.818   77.8472   127.187    32.9995
 MEMORY USED   = 2400   2400   2400   2800   7800   200
                                                               29.739
 Z = 340997
```

Over one run cycle of the subsystem, the *I/O* time per file becomes

```
                                                                 19.033
  IOTIME (SEC.) = 11326.1  11326.1  1096.03  1790.69  116.149
TOTAL
```

Percent deviations from the optimum are computed to be

$\Delta'' =$

```
                                                 -1.73   217
PERCENT TIME DIFF FROM OPTIMAL = 18.3   18.3   ¯34.7   ¯22.1
PERCENT LENG DIFF FROM OPTIMAL = 18.3   18.3   ¯34.7   ¯22.1
PERCENT TOTAL TIME DIFF FROM OPTIMAL = 10.4
PERCENT TOTAL LENG DIFF FROM OPTIMAL = 10.8     -1.73   217
```

A brief look at the sample program output in appendices A and B will show a distinct difference between using local and global optimization. Local optimization for program B yields $X_1 = 31.79$, $X_5 = 12.31$, whereas global optimization yields $X_1 = 12.24$, $X_5 = 35.21$ Thus the two approaches are not always compatible.

It should be emphasized that these percentage reductions pertain to *I/O TIME*, and does not necessarily entail the same corresponding reduction in throughput time, since the latter is dependent also on processing time, overlapped operations, and many other related system parameters. Nevertheless, this small example illustrates the power and simplicity of this optimization technique. Furthermore, it can be expected that the larger the subsystem and the more disparity there is among file sizes, record sizes, program run frequencies, and memory constraints, the more substantial one can expect the gains to be from such a system approach.

```
C     NONLINEAR PROGRAM TO SOLVE FOR OPTIMAL BLOCKING FACTORS
C     FOR TAPE FILES IN A SUBSYSTEM OF TWO OR MORE
C     INTERRELATED PROGRAMS.
C
C     FORMAT FOR INPUT DATA:
C     CARD 1      (RIGHT-JUSTIFIED INTEGERS)
C           COLS  1- 5   NO. OF TAPE FILES IN SUBSYSTEM
C           COLS  6-10   NO. OF CONSTRAINTS
C           COLS 11-15   LIMIT NO. OF ITERATIONS
C           COLS 16-20   START PRINTING FROM THIS ITERATION
C           COLS 21-25   STOP PRINTING AT THIS ITERATION
C           COLS 26-30   PRINT ITERATIONS OF THIS MULTIPLE
C     CARDS 2 TO I       INITIAL FEASIBLE SOLUTION
C           10 COLS PER FIELD, 8 FIELDS PER CARD (DECIMAL)
C     CARDS I+1 TO J     NO. OF ALTERNATE TAPE BUFFERS
C           2 COLS PER FIELD, 40 FIELDS PER CARD (INTEGER)
C     CARDS J+1 TO K     RECORD SIZES
C           10 COLS PER FIELD, 8 FIELDS PER CARD (DECIMAL)
C     CARDS K+1 TO L     FILE SIZES
C           10 COLS PER FIELD, 8 FIELDS PER CARD (DECIMAL)
C     CARDS L+1 TO M     FREQUENCIES OF PROGRAM RUNS
C           10 COLS PER FIELD, 8 FIELDS PER CARD (DECIMAL)
C     CARDS M+1 TO N     MEMORY CONSTRAINTS MATRIX (BY ROW)
C           10 COLS PER FIELD, 8 FIELDS PER CARD (DECIMAL)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION CL(6,9), VL(9), Y(9)
      DIMENSION FREQ(6), FSIZE(6), RSIZE(6), IALT(6)
C
C     INPUT MATRIX DIMENSIONS, LIMIT NO. OF ITERATIONS,
C     PRINTING PARAMETERS AND CONVERGENCE TOLERANCE.
C
      READ (5,100) NV,NC,LIMIT,MFROM,MTO,MUL,EPS
      WRITE (6,101) NV, NC, LIMIT, EPS
C
C     INPUT INITIAL FEASIBLE SOLUTION
C
      READ (5,110) (Y(I),I=1,NV)
C
C     INPUT NO. OF ALTERNATE TAPE BUFFERS
C
      READ (5,10) (IALT(I),I=1,NV)
C
C     INPUT RECORD SIZES
C
```

```
      READ (5,110) (RSIZE(I),I=1,NV)
C
C      INPUT FILE SIZES
C
      READ (5,110) (FSIZE(I),I=1,NV)
C
C      INPUT RUN FREQUENCIES
C
      READ (5,110) (FREQ(I),I=1,NV)
C
C      INPUT MEMORY CONSTRAINTS MATRIX
C
      WRITE (6,111)
      DO 50 J=1,NC
      READ (5,110) ((CL(I,J),I=1,NV),VL(J))
50    WRITE (6,112) ((CL(I,J),I=1,NV),VL(J))
      WRITE (6,201) (I,Y(I),I=1,NV)
      CALL GPROJ (CL,VL,Y,F,NV,NC,EPS,LIMIT,MFROM,MTO,MUL,IER,
     *              FREQ,FSIZE,IT)
      IF(IER.EQ.0) GO TO 60
      WRITE(6,204) IT,(I,Y(I),I=1,NV)
      WRITE(6,205) (I,IALT(I),I,RSIZE(I),I,FSIZE(I),
     *                  I,FREQ(I),I=1,NV)
      WRITE (6,206) F
      WRITE (6,250) IER
      GO TO 70
60    WRITE(6,209) IT,(I,Y(I),I=1,NV)
      WRITE(6,205) (I,IALT(I),I,RSIZE(I),I,FSIZE(I),
     *                  I,FREQ(I),I=1,NV)
      WRITE (6,220) F
10    FORMAT (40I2)
100   FORMAT (6I5,F10.6)
101   FORMAT ('1',/,'ONO. OF TAPE FILES              =',I4,
     */' NO. OF MEMORY CONSTRAINTS =',I4,
     */' LIMIT NO. OF ITERATIONS    =',I4,
     */' CONVERGENCE TOLERANCE      =',E8.1)
110   FORMAT (8F10.2)
111   FORMAT ('OMEMORY CONSTRAINTS MATRIX (LHS.GE.RHS) :',/)
112   FORMAT (1X,16F8.0)
201   FORMAT ('OINITIAL FEASIBLE SOLUTION :',
     *//,(' X(',I2,') =',F6.2))
204   FORMAT('ONO CONVERGENCE AFTER ',I5,'  ITERATIONS :',
     *//,(' X(',I2,') =',F6.2))
205   FORMAT('OALT. BUFFERS:',T17,'RECORD SIZES:',
     *T32,'FILE SIZES:',T50,'RUN FREQ:',
     *//(' A(',I2,') =',I2,T17,'R(',I2,') =',F5.0,T32,
     *'F(',I2,') =',F8.0,T50,'FREQ(',I2,') =',F7.2))
206   FORMAT('OVALUE OF OBJECTIVE FUNCTION  Z =',F9.0)
```

```
209    FORMAT ('OCONVERGED TO OPTIMAL SOLUTION AFTER ',I5,
      *'  ITERATIONS :',//,(' X(',I2,') =',F6.2))
220    FORMAT ('OMINIMUM VALUE OF OBJECTIVE FUNCTION  Z =',
      *F9.0)
250    FORMAT ('OERROR CODE =',I4)
70     CONTINUE
       STOP
       END
```

```
SUBROUTINE GPROJ

PURPOSE:
   TO MINIMIZE NONLINEAR OBJECTIVE FUNCTIONS WITH LINEAR
   CONSTRAINTS USING THE GRADIENT PROJECTION METHOD.

USAGE:
   CALL GPROJ (CL,VL,Y,F,NV,NC,EPS,LIMIT,MFROM,MTO,MUL,IER,
               FREQ,FSIZE,IT)

DESCRIPTION OF PARAMETERS:
   CL     - INPUT COEFFICIENT MATRIX (DIMENSION NV*NC) OF THE
            LINEAR CONSTRAINTS IN THE FORM CL*Y.GE.VL (REAL*8).
   VL     - INPUT VECTOR OF R.H.S. OF LINEAR CONSTRAINTS OF
            DIMENSION NC (REAL*8).
   Y      - VECTOR OF DIMENSION NV CONTAINS THE INITIAL ARGUMEN
            WHERE THE ALGORITHM STARTS. ON RETURN, Y CONTAINS
            THE ARGUMENT CORRESPONDING TO THE COMPUTED MINIMUM
            FUNCTION VALUE (REAL*8).
   F      - CONTAINS MINIMUM FUNCTION VALUE ON RETURN
            (I.E. F=F(Y)) (REAL*8).
   NV     - NUMBER OF VARIABLES (INTEGER*4).
   NC     - NUMBER OF CONSTRAINTS (INTEGER*4).
   EPS    - TEST VALUE FOR ZERO APPROXIMATED BY ROUNDOFF ERROR.
            SUITABLE VALUES IN RANGE 10**-4 TO 10**-6 (REAL*8).
   LIMIT  - MAXIMUM NUMBER OF ITERATIONS. SUGGESTED LIMIT IS
            LIMIT = NC (INTEGER*4).
   MFROM  - START PRINTING FROM THIS ITERATION.
   MTO    - STOP PRINTING AT THIS ITERATION.
   MUL    - PRINT ITERATIONS OF THIS MULTIPLE.
   IER    - ERROR CODE (INTEGER*4).
            IER= 0 : NO ERROR
            IER= J : CONSTRAINT J INVALID OR MISSING
            IER=-1 : FEASIBLE REGION UNBOUNDED (I.E. NC IS NOT
                     GREATER THEN NV)
            IER=-2 : INITIAL VALUE OF Y IS NOT FEASIBLE
            IER=-3 : CONSTRAINTS LINEARLY DEPENDENT (I.E. Q IS
                     GREATER THAN NV)
            IER=-4 : NO CONVERGENCE IN LIMIT ITERATIONS
   FREQ   - FREQUENCY EACH TAPE FILE IS PROCESSED IN RUN CYCLE
            OF SUBSYSTEM. (CAN BE FRACTIONAL)
   FSIZE  - AVERAGE NO. OF RECORDS PER TAPE FILE.
   IT     - ITERATION COUNT
   NVMAX  = LARGEST VALUE OF NV TO BE PROCESSED
   MAX    = NVMAX+1
```

```
NONDUMMY DIMENSIONS YY(MAX),DF(MAX),Z(MAX),VQ(MAX),H(MAX),
                    CQ(MAX*MAX),CQT(MAX*MAX),T(MAX*MAX),
                    TI(MAX*MAX),R(MAX*MAX),PQ(MAX*MAX)
```

```
ROUTINES AND FUNCTION SUBPROGRAMS REQUIRED:
MEQAL  - DSOS:DSOLIB  - MATRIX EQUATE
MADD   - DSOS:DSOLIB  - MATRIX ADD
MMPY   - DSOS:DSOLIB  - MATRIX MULTIPLY   (NONDUMMY DIMENSION
MINVR  - DSOS:DSOLIB  - MATRIX INVERSION  (NONDUMMY DIMENSION
DOT    - DSOS:DSOLIB  - VECTOR DOT PRODUCT
MTRAS  - SUPPLIED     - MATRIX TRANSPOSE
FUNCT  - USER-WRITTEN - A SUBROUTINE CONCERNING THE FUNCTION
                        TO BE MINIMIZED, MUST BE OF THE FORM
                           SUBROUTINE FUNCT (N,ARG,VAL,GRAD)
                        AND MUST SERVE THE FOLLOWING PURPOSE:
                        TO EACH N-DIMENSIONAL ARGUMENT VECTOR
                        A FUNCTION VALUE AND GRADIENT VECTOR
                        MUST BE COMPUTED, AND ON RETURN
                        STORED IN VAL AND GRAD RESPECTIVELY
                        (N.B. ARG, VAL, GRAD ARE REAL*3).
```

REMARKS:

(1) INPUT MATRIX CL ASSUMED TO BE STORED COLUMNWISE IN NV*NC SUCCESIVE LOCATIONS.

(2) BOTH CL AND VL ARE MODIFIED AS THE COEFFICIENTS ARE NORMALIZED IN THE ALGORITHM.

(3) NC MUST BE GREATER THAN NV.

(4) INITIAL VALUE OF Y IS REQUIRED TO LIE IN THE FEASIBLE REGION DEFINED BY THE LINEAR CONSTRAINTS.

(5) ALL ROUTINES USED IN CONJUNCTION WITH THIS SUBROUTINE MUST HAVE REAL VARIABLES IN DOUBLE PRECISION FORM (I.E. REAL*8). THIS CAN EASILY BE DONE BY USE OF THE FORTRAN STATEMENT

```
IMPLICIT REAL*8(A-H,O-Z)
```

AT THE BEGINNING OF EACH ROUTINE.

REFERENCES:

(1) KIRK, D.E., 'OPTIMAL CONTROL THEORY - AN INTRODUCTION', PRENTICE-HALL, 1970, PP. 373-393.

(2) ROSEN, J.B., 'THE GRADIENT PROJECTION METHOD FOR NONLINEAR PROGRAMMING PART 1, LINEAR CONSTRAINTS', J.SIAM, VOL.8, 1960, PP. 181-217.

```
      SUBROUTINE GPROJ (CL,VL,Y,F,NV,NC,EPS,LIMIT,
     *                  MFROM,MTO,MUL,IER,FREQ,FSIZE,IT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION CL(NV,NC),VL(NC),Y(NV)
      DIMENSION FREQ(NV), FSIZE(NV)
      DIMENSION YY(7),DF(7),Z(7),VQ(7),H(7),CQ(49),CQT(49),
     *          T(49),TI(49),R(49),PQ(49)
      INTEGER*4 Q
      INTEGER*2 H
      REAL*8 INF/7.D75/
C
C     INITIALIZE VARIABLES
C
      IF(NC.LE.NV)GO TO 91
      IT=0
      Q=0
C
C     NORMALIZE LINEAR CONSTRAINTS
C
      DO 2 J=1,NC
      S=DOT(CL(1,J),CL(1,J),NV)
      IF(DABS(S).LT.EPS)GO TO 97
      IF(DABS(S-1.).LT.EPS)GO TO 2
      S=1./DSQRT(S)
      DO 1 I=1,NV
    1 CL(I,J)=S*CL(I,J)
      VL(J)=S*VL(J)
    2 CONTINUE
C
C     DETERMINE ACTIVE CONSTRAINTS AND FEASIBILITY OF POINT Y
C
      DO 5 J=1,NC
      S=DOT(CL(1,J),Y,NV)-VL(J)
      IF(DABS(S).GE.EPS)GO TO 4
      Q=Q+1
      H(Q)=J
      GO TO 5
    4 IF(S.LT.0.)GO TO 92
    5 CONTINUE
C
C     DETERMINE PROJECTION MATRIX PQ
C
   10 IF(Q.GT.NV+1)GO TO 93
      IT=IT+1
      IF(IT.GT.LIMIT)GO TO 94
```

```
   11 DO 13 I=1,NV
      DO 13 J=1,NV
      IJ=I+(J-1)*NV
      IF(I.EQ.J)GO TO 12
      PQ(IJ)=0.
      GO TO 13
   12 PQ(IJ)=1.
   13 CONTINUE
      IF(Q.EQ.0)GO TO 15
      DO 14 I=1,Q
      J=H(I)
      IJ=1+(I-1)*NV
      CALL MEQAL(CQ(IJ),CL(1,J),NV,1,1)
   14 VQ(I)=VL(J)
      CALL MTRAS(CQT,CQ,NV,Q)
      CALL MMPY(T,CQT,CQ,Q,NV,Q)
      CALL MINVR(TI,T,DET,Q)
      CALL MMPY(R,TI,CQT,Q,Q,NV)
      CALL MMPY(T,CQ,R,NV,Q,NV)
      CALL MADD(PQ,PQ,T,NV,NV,-1)
C
C     CALCULATE GRADIENT PROJECTION Z
C
   15 CALL FUNCT(NV,Y,F,DF,FREQ,FSIZE)
      CALL MEQAL(DF,DF,NV,1,-1)
      CALL MMPY(Z,PQ,DF,NV,NV,1)
      ZNORM=DSQRT(DOT(Z,Z,NV))
C
C     CALCULATE MATRIX R AND RQ
C
      RQ=-INF
      IF(Q.EQ.0)GO TO 18
      CALL MMPY(T,CQT,DF,Q,NV,1)
      CALL MMPY(R,TI,T,Q,Q,1)
      DO 17 I=1,Q
      IF(RQ.GE.R(I))GO TO 17
      RQ=R(I)
      K=I
   17 CONTINUE
C
C     TEST FOR CONSTRAINED MINIMUM
C
   18 IF(ZNORM.GT.EPS)GO TO 20
      IF(RQ.LE.EPS)GO TO 98
      GO TO 25
C
C     CALCULATE BETA
C
```

```
   20 IF(Q.EQ.0)GO TO 26
      BETA=0.
      DO 22 I=1,Q
      S=0.
      DO 21 J=I,Q
      IJ=I+(J-1)*Q
   21 S=S+DABS(TI(IJ))
      IF(BETA.LT.S)BETA=S
   22 CONTINUE
      IF(RQ.LE.BETA)GO TO 26
C
C     DROP HYPERPLANE H(K) FROM ACTIVE CONSTRAINTS
C
   25 H(K)=H(Q)
      Q=Q-1
      GO TO 11
C
C     FORM NORMALIZED GRADIENT PROJECTION Z
C
   26 S=1./ZNORM
      DO 27 I=1,NV
   27 Z(I)=S*Z(I)
C
C     DETERMINE MAXIMUM POSSIBLE STEP SIZE (STEPM)
C
      STEPM=INF
      DO 30 J=1,NC
      IF(Q.EQ.0)GO TO 29
      DO 28 I=1,Q
      IF(J.EQ.H(I))GO TO 30
   28 CONTINUE
   29 S=DOT(CL(1,J),Z,NV)
      IF(DABS(S).LT.EPS)GO TO 30
      STEP=(VL(J)-DOT(CL(1,J),Y,NV))/S
      IF(STEP.LE.0.OR.STEP.GT.STEPM)GO TO 30
      STEPM=STEP
      M=J
   30 CONTINUE
C
C     DETERMINE TENTATIVE NEXT POINT YY
C
   35 DO 36 I=1,NV
   36 YY(I)=Y(I)+STEPM*Z(I)
      CALL FUNCT(NV,YY,F,DF,FREQ,FSIZE)
      CALL MEQAL(DF,DF,NV,1,-1)
      IF(DOT(DF,Z,NV).GE.-EPS)GO TO 45
C
C     FIND NEXT POINT YY BY BISECTION SEARCH
```

```
   40 TH=STEPM/2.
      STEPM=TH
   41 DO 42 I=1,NV
   42 YY(I)=Y(I)+STEPM*Z(I)
      CALL FUNCT(NV,YY,F,DF,FREQ,FSIZE)
      CALL MEQAL(DF,DF,NV,1,-1)
      S=DOT(DF,Z,NV)
      IF(DABS(S).LT.EPS)GO TO 46
      TH=TH/2.
      STEPM=STEPM+DSIGN(1.D0,S)*TH
      GO TO 41
C
C     ADD HYPERPLANE M TO ACTIVE CONSTRAINTS
C
   45 Q=Q+1
      H(Q)=M
C
C     UPDATE Y AND START NEXT ITERATION
C
   46 CALL MEQAL(Y,YY,NV,1,1)
C
C     PRINT INTERMEDIATE ITERATIONS
C
      ML=IT/MUL*MUL
      IF(IT.LT.MFROM.OR.IT.GT.MTO.OR.IT.NE.ML) GO TO 10
      WRITE (6,1000) IT, (I, Y(I),I=1,NV)
 1000 FORMAT ('0SOLUTION AFTER ',I5,'  ITERATIONS :',
     *//,(' X(',I2,') =',F6.2))
      GO TO 10
C
C     RETURN ERROR CODE
C
   91 IER=-1
      GO TO 99
   92 IER=-2
      GO TO 99
   93 IER=-3
      GO TO 99
   94 IER=-4
      GO TO 99
   97 IER=J
      GO TO 99
   98 IER=0
   99 RETURN
      END
```

```
      SUBROUTINE MADD (A,B,C,NR,NC,KS)
C
C     IF KS.LT.0, SUBROUTINE MADD SETS A EQUAL TO B.
C     IF KS.GE.0, SUBROUTINE MADD SETS A EQUAL TO B PLUS C.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(NR,NC),B(NR,NC),C(NR,NC)
      IF(KS.GE.0) GO TO 200
      DO 150 I=1,NR
      DO 150 J=1,NC
  150 A(I,J)=B(I,J)-C(I,J)
      RETURN
  200 DO 250 I=1,NR
      DO 250 J=1,NC
  250 A(I,J)=B(I,J)+C(I,J)
      RETURN
      END
```

```
      SUBROUTINE MEQAL (A,B,NR,NC,KS)
C
C     IF KS.LT.0, SUBROUTINE MEQAL SETS A EQUAL TO MINUS B.
C     IF KS.GE.0, SUBROUTINE MEQAL SETS A EQUAL TO B.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(NR,NC),B(NR,NC)
      IF(KS.GE.0) GO TO 200
      DO 150 I=1,NR
      DO 150 J=1,NC
  150 A(I,J)=-B(I,J)
      RETURN
  200 DO 250 I=1,NR
      DO 250 J=1,NC
  250 A(I,J)=B(I,J)
      RETURN
      END
```

```
      SUBROUTINE MMPY (A,B,C,NRA,NCB,NCA)
C
C     SUBROUTINE MMPY SETS MATRIX A EQUAL TO MATRIX B
C     POST-MULTIPLIED BY MATRIX C.
C     NRA = NUMBER OF ROWS IN A,B
C     NCB = NUMBER OF COLUMNS IN B, ROWS IN C
C     NCA = NUMBER OF COLUMNS IN A,C
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1024),B(1024),C(1024),D(1024)
      DO 100 I=1,NRA
      DO 100 J=1,NCA
      MD=(J-1)*NRA+I
      D(MD)=0.0
      DO 100 K=1,NCB
      MA=(K-1)*NRA+I
      MB=(J-1)*NCB+K
  100 D(MD)=D(MD)+B(MA)*C(MB)
      NME=NRA*NCA
      DO 200 I=1,NME
  200 A(I)=D(I)
      RETURN
      END
```

```
      SUBROUTINE MTRAS (R,A,M,N)
C
C     SUBROUTINE MTRAS TRANSPOSES MATRIX A.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(M,N),R(N,M)
      DO 1 I=1,M
      DO 1 J=1,N
    1 R(J,I)=A(I,J)
      RETURN
      END
```

```
      SUBROUTINE MINVR (A,B,DET,N)
C
C     SUBROUTINE MINVR COMPUTES THE INVERSE AND DETERMINANT
C     OF MATRIX B.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(N,N),B(N,N)
      DIMENSION PIVOT(32),IPIVOT(32),INDEX(32,2)
      EQUIVALENCE (IROW,JROW),(ICOLUM,JCOLUM),(AMAX,T,SWAP)
C
C     INITIALIZE VARIABLES
C
      DET=1.0
      DO 40 I=1,N
      IPIVOT(I)=0
      DO 40 J=1,N
   40 A(I,J)=B(I,J)
C
C     PERFORM INVERSION ROUTINE
C
      DO 600 I=1,N
C
C     SEARCH FOR PIVOT ELEMENT PIVOT(I).
C
      AMAX=0.0
      DO 110 J=1,N
      IF(IPIVOT(J).EQ.1) GO TO 110
      DO 100 K=1,N
      IF(IPIVOT(K)-1)80,100,1000
   80 IF(DABS(AMAX).GE.DABS(A(J,K))) GO TO 100
      IROW=J
      ICOLUM=K
      AMAX=A(J,K)
  100 CONTINUE
  110 CONTINUE
      IPIVOT(ICOLUM)=IPIVOT(ICOLUM)+1
C
C     INTERCHANGE ROWS TO PLACE PIVOT ELEMENTS ON DIAGONAL.
C
      IF(IROW.EQ.ICOLUM) GO TO 260
      DET=-DET
      DO 200 L=1,N
      SWAP=A(IROW,L)
      A(IROW,L)=A(ICOLUM,L)
  200 A(ICOLUM,L)=SWAP
```

```
  260 INDEX(I,1)=IROW
      INDEX(I,2)=ICOLUM
      PIVOT(I)=A(ICOLUM,ICOLUM)
      DET=DET*PIVOT(I)
C
C     NORMALIZE PIVOT ROW ON PIVOT ELEMENT.
C
      A(ICOLUM,ICOLUM)=1.0
      DO 350 L=1,N
  350 A(ICOLUM,L)=A(ICOLUM,L)/PIVOT(I)
C
C     REDUCE NON-PIVOT ROWS.
C
      DO 550 L1=1,N
      IF(L1.EQ.ICOLUM) GO TO 550
      T=A(L1,ICOLUM)
      A(L1,ICOLUM)=0.0
      DO 450 L=1,N
  450 A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
  550 CONTINUE
  600 CONTINUE
C
C     INTERCHANGE COLUMNS.
C
      DO 710 I=1,N
      L=N+1-I
      IF(INDEX(L,1).EQ.INDEX(L,2)) GO TO 710
      JROW=INDEX(L,1)
      JCOLUM=INDEX(L,2)
      DO 700 K=1,N
      SWAP=A(K,JROW)
      A(K,JROW)=A(K,JCOLUM)
      A(K,JCOLUM)=SWAP
  700 CONTINUE
  710 CONTINUE
 1000 RETURN
      END
```

```
      FUNCTION DOT (X,Y,NN)
C
C     FUNCTION DOT CALCULATES THE DOT-PRODUCT OF
C     NN-DIMENSIONAL VECTORS X AND Y.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(NN),Y(NN)
      DPR=0.0
      DO 100 I=1,NN
  100 DPR=DPR+X(I)*Y(I)
      DOT=DPR
      RETURN
      END
```

```
      SUBROUTINE FUNCT (N,X,VAL,GRAD,FREQ,FSIZE)
C
C     SUBROUTINE FUNCT COMPUTES THE VALUE OF THE
C     OBJECTIVE FUNCTION AND ITS PARTIAL DERIVATIVES
C     WITH RESPECT TO EACH VARIABLE AT POINT X.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N), GRAD(N), FREQ(N), FSIZE(N)
      VAL=0.
      DO 10 I=1,N
      VAL=VAL+FREQ(I)*FSIZE(I)/X(I)
10    GRAD(I)=-FREQ(I)*FSIZE(I)/X(I)**2
      RETURN
      END
```

SAMPLE OUTPUT :

```
NO. OF TAPE FILES            =   6
NO. OF MEMORY CONSTRAINTS =   9
LIMIT NO. OF ITERATIONS     = 500
CONVERGENCE TOLERANCE       = 0.1D-02

MEMORY CONSTRAINTS MATRIX (LHS.GE.RHS) :

   -300.    -300.    -160.    -140.       0.       0. -10000.
   -300.       0.       0.       0.    -200.       0. -12000.
      0.       0.       0.       0.    -200.    -200.  -8000.
      1.       0.       0.       0.       0.       0.      0.
      0.       1.       0.       0.       0.       0.      0.
      0.       0.       1.       0.       0.       0.      0.
      0.       0.       0.       1.       0.       0.      0.
      0.       0.       0.       0.       1.       0.      0.
      0.       0.       0.       0.       0.       1.      0.

INITIAL FEASIBLE SOLUTION :

X( 1) = 10.00
X( 2) = 10.00
X( 3) = 10.00
X( 4) = 10.00
X( 5) = 10.00
X( 6) = 10.00

SOLUTION AFTER   100  ITERATIONS :

X( 1) = 12.25
X( 2) = 12.25
X( 3) =  7.11
X( 4) = 10.80
X( 5) = 24.43
X( 6) = 10.76

SOLUTION AFTER   200  ITERATIONS :

X( 1) = 12.24
X( 2) = 12.24
X( 3) =  7.14
X( 4) = 10.80
X( 5) = 34.72
X( 6) =  5.28

SOLUTION AFTER   300  ITERATIONS :

X( 1) = 12.24
X( 2) = 12.24
X( 3) =  7.14
X( 4) = 10.80
X( 5) = 35.03
X( 6) =  4.97
```

```
SOLUTION AFTER   400  ITERATIONS :

X( 1) = 12.24
X( 2) = 12.24
X( 3) =  7.14
X( 4) = 10.80
X( 5) = 35.16
```

```
NO CONVERGENCE AFTER   501  ITERATIONS :

X( 1) = 12.24
X( 2) = 12.24
X( 3) =  7.14
X( 4) = 10.80
X( 5) = 35.21
X( 6) =  4.79

ALT. BUFFERS:  RECORD SIZES:  FILE SIZES:        RUN FREQ:

A( 1) = 2      R( 1) = 150.   F( 1) =  50000.    FREQ( 1) =24.2
A( 2) = 2      R( 2) = 150.   F( 2) =  50000.    FREQ( 2) =24.2
A( 3) = 2      R( 3) =  80.   F( 3) =  10000.    FREQ( 3) =22.0
A( 4) = 2      R( 4) =  70.   F( 4) =  20000.    FREQ( 4) =22.0
A( 5) = 2      R( 5) = 100.   F( 5) =   5000.    FREQ( 5) = 5.5
A( 6) = 2      R( 6) = 100.   F( 6) =    500.    FREQ( 6) = 1.0

VALUE OF OBJECTIVE FUNCTION  Z =  270516.

ERROR CODE =  -4
```

BIBLIOGRAPHY

A. Computer Systems Optimization

[1] Abate, J., and Dubner, H. Optimizing the performance of a drum-like storage. IEEE Trans. Computers, Vol. C-18, No. 9, (Nov. 1969), pp. 992-997.

[2] Abate, J., Dubner, H., and Weinberg, S.B. Queueing analysis of the IBM/2314 disk storage facility. Vol. 11, No. 10, (Oct. 1968), pp. 577-589.

[3] Arora, S.R., and Gallo, A. Optimal sizing, loading and re-loading in a multi-level memory hierarchy system. AFIPS-SJCC Proceedings, Vol. 38, (1971), pp. 337-344.

[4] Belady, L.A., and Kuehner, C.J. Dynamic space-sharing in computer systems. C.ACM, Vol. 12, No. 5, (May 1969), pp. 282-288.

[5] Black, N.A. Optimum merging from mass storage. C.ACM, Vol. 13, No. 12, (Dec. 1970), pp. 745-749.

[6] Cantrell, H.N., and Ellison, A.L. Multiprogramming system performance measurement and analysis. AFIPS-SJCC Proceedings, Vol. 32, (May 1968), pp. 213-222.

[7] Chang, W., and Wong, D.J. Computer channel interference analysis. IBM Systems J., Vol. 4, No. 2, (1965), pp. 162-170.

[8] Chapin, N. Common file organization techniques compared. AFIPS-FJCC Proceedings, Vol. 35, (1969), pp. 413-422.

[9] Chu, W.W. Optimal file allocation in a multiple computer system. IEEE Trans. Computers, Vol. C-18, No. 10, (Oct. 1969), pp. 885-889.

[10] Coffman, E.G. Studying multiprogramming systems. Datamation, (June 1967), pp. 47-56.

[11] Collmeyer, A.J., and Shemer, J.E. Analysis of retrieval performance for selected file organization techniques. AFIPS-FJCC Proceedings, Vol. 37, (1970), pp. 201-207.

[12] Daley, R.C., and Neumann, P.G. A general-purpose file structure for secondary storage. AFIPS-FJCC Proceedings, Vol. 27, Part 1, (1965), pp. 213-229.

[13] Denning, P.J. Effects of scheduling on file memory operations. AFIPS-SJCC Proceedings, Vol. 30, (1967), pp. 9-21.

[14] Dijkstra, E.W. The structure of the THE-multiprogramming system. C.ACM, Vol. 11, No. 5,(May 1968),pp.341-346.

[15] Drummond, Jr., M.E. A perspective on system performance evaluation. IBM Systems J., Vol. 8, No. 4, (1969), pp. 252-264.

[16] Ferguson, D.E. Buffer allocation in merge-sorting. C.ACM, Vol. 14, No. 7, (July 1971), pp. 476-478.

[17] Frank, H. Analysis and optimization of disk storage devices for time-sharing systems. J.ACM, Vol. 16, No. 4, (Oct. 1969), pp. 602-620.

[18] Ghosh, S.P., and Senko, M.E. File organization: on the selection of random access index points for sequential files. J.ACM, Vol. 16, No. 4, (Oct. 1969), pp. 569-579.

[19] Hellerman, H., and Smith, Jr., H.J. Throughput analysis of some idealized input, output, and compute overlap configurations. Computing Surveys (ACM), Vol. 2, No. 2, (June 1970), pp. 111-118.

[20] Hess, H. A comparison of disks and tapes. C.ACM, Vol. 6, No. 10, (Oct. 1963), pp. 634-638.

[21] Howard, P.C. Optimizing performance in a multiprogramming system. Datamation, (Jan. 1969), pp. 65-67.

[22] Lowe, T.C. The influence of data-base characteristics and usage on direct-access file organization. J.ACM, Vol. 15, No. 4, (Oct. 1968), pp. 535-548.

[23] Madnick, S.E., and Alsop, J.W. A modular approach to file system design. AFIPS-SJCC Proceedings, Vol. 34, (1969), pp. 1-13.

[24] Manocha, T., Martin, W.L., and Stevens, K.W. Performance evaluation of direct access storage devices with a fixed head per track. AFIPS-SJCC Proceedings, Vol. 38, (1971), pp. 309-317.

[25] Pinkerton, T.B. Performance monitoring in a time-sharing system. C.ACM, Vol. 12, No. 11, (Nov. 1969).

[26] Randell, B., and Kuehner, C.J. Dynamic storage allocation systems. C.ACM, Vol. 11, No. 5, (May 1968), pp. 297-306.

[27] Ricci, J.M. Precision magnetic tape. Datamation, (Oct. 1966).

[28] Shemer, J.E., and Heying, D.W. Performance modeling and empirical measurements in a system designed for batch and time-sharing users. AFIPS-FJCC Proceedings, Vol. 35, (1969), pp. 17-26.

[29] Silver, E.A., Loss, A.L., and Black, F. A quantitative rule for the use of resources in a multiprogrammed computer system. INFOR J., Vol. 9, No. 2, (July 1971), pp. 96-110.

[30] Skinner, C.E., and Asher, J.R. Effects of storage contention on system performance. IBM Systems J., Vol. 8, No. 4, (1969), pp. 319-334.

[31] Stimler, S., and Brons, K.A. A methodology for calculating and optimizing real-time system performance. C.ACM, Vol. 11, No. 7, (July, 1968), pp. 509-516.

[32] Walker, E.S. Optimization of tape operations. Software Age, (Aug./Sept. 1970), pp. 16-17.

[33] Waters, S.J. Blocking sequentially processed magnetic files. Computer J., Vol. 14, No. 2, (May 1971), pp. 109-112.

[34] Wilkes, M.V. A model for core space allocation in a time-sharing system. AFIPS-SJCC Proceedings, Vol. 34, (1969), pp. 265-271.

[35] Wolman, E. A fixed optimum cell-size for records of various lengths. J.ACM, Vol. 12, No. 1, (Jan. 1965), pp. 53-70.

[36] Woodrum, L.J. A model of floating buffering. IBM Systems J., Vol. 9, No. 2, (1970), pp. 118-144.

B. Operations Research

[37] Ackoff, R.L. Progress in Operations Research, Vol. I, John Wiley & Sons, Inc., 1961.

[38] Ackoff, R.L., and Sasieni, M.W. Fundamentals of Operations Research. John Wiley & Sons, Inc., 1968.

[39] Batchelor, J.H. OR, an Annotated Bibliography. St. Louis University Press, St. Louis, Missouri, Vol. 1, 1959, Vol. 2, 1962.

[40] Churchman, C.W. The Systems Approach. Delacorte Press, 1968.

[41] ----. Managerial acceptance of scientific recommendations. California Management Review, (Fall 1964).

[42] Churchman, C.W., Ackoff, R.L., and Arnoff, E.L. Introduction to Operations Research. John Wiley & Sons, Inc., 1957.

[43] Churchman, C.W., and Schainblatt, A.H. The researcher and the manager: a dialectic of implementation. Management Science, Vol. 11, No. 4, (Feb. 1965), pp. B69-B87.

[44] Coch, L., and French, Jr., J.R.P. Overcoming resistence to change. Human Relations, Vol. 1, No. 4, (1948), p. 512.

[45] Creelman, G.D., and Wallen, R.W. The place of psychology in operations research. Operations Research, Vol. 6, No. 1, (Jan-Feb. 1958), p. 119.

[46] Everett, H. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Operations Research, Vol. 11, No. 3, (May-June 1963), pp. 399-417.

[47] Hertz, D.B. New Power for Management-Computer Systems and Management Science. McGraw-Hill, 1969.

[48] Hertz, D.B., and Eddison, R.T. Progress in Operations Research, Vol. II, John Wiley & Sons, 1964.

[49] Hillier, F.S., and Lieberman, G.J. Introduction to Operations Research. Holden-Day, Inc., 1967.

[50] Hitch, C.S. Sub-optimization in operations problems. Operations Research, 1, (1953), PP. 87-99.

[51] Hovland, C.I. (ed.) The Order of Presentation in Persuasion. Yale Studies in Attitude and Communication, Vol.1, Yale Univ. Press, New Haven, 1960.

[52] Huysmans, J.H.B.M. The Implementation of Operations Research. John Wiley & Sons, 1970.

[53] Malcolm, D.G. On the need for improvement in implementation of OR. Management Science, Vol. 11, No. 4, (Feb. 1965), pp. 48-58.

[54] McKinsey and Co., Inc. A Limited Survey of Industrial Progress in OR. Reinhold Publishing Co., N.Y., 1965.

[55] Morse, P.M. OR - what is it ? Proceedings of the First Seminar in Operations Research, Case Institute of Technology, (Nov. 1951).

[56] Oldaker, R.B. A Survey of OR Organizations in Selected U.S. Corporations. Unpublished Master's thesis, Graduate School of Business Administration, Univ. of California, Berkeley, (Jan. 1966).

[57] Pugh, G.E. Lagrange multipliers and the optimal allocation of defense resources. Operations Research, Vol. 12, No. 4, (July-Aug. 1964), pp. 543-567.

[58] Schumacher, C.C., and Smith, B.E. A sample survey of industrial operations research activities II. Operations Research, Vol. 13, No. 6,(Nov.-Dec. 1965), pp. 1023-1027.

[59] Stillson, P. Implementation of problems in OR. Operations Research, Vol. 11, No. 1,(Jan-Feb. 1963), p. 141.

[60] Turban, E. How companies are using OR techniques, Management Review, Vol. 59, No. 3, (Mar. 1970), pp. 10-14.

[61] Wagner, H.M. Principles of Operations Research With Applications to Managerial Decisions. Prentice-Hall, 1969.

C. Non-linear Programming

[62] Abadie, J. (ed.) Non-linear Programming. John Wiley & Sons, 1967.

[63] Arrow, K., and Hurwicz, L. Gradient methods for constrained maxima. Operations Research, Vol. 5, No. 2, (April 1957), pp. 258-265.

[64] Arrow, K., Hurwicz, L., and Uzawa, H. Studies in Linear and Non-linear Programming. Stanford Univ. Press, 1958.

[65] Balas, E. Extension de l'algorithme additif a la programmation en nombres entiers et a la programmation non-lineaire. Comptes Rendue de l'Academie des Sciences (Paris), 258, (1964), p. 5136-5139.

[66] Bellman, R. Dynamic programming and the numerical solution of variational problems. Operations Research, Vol. 5, No. 2, (April 1957), pp. 277-288.

[67] Bellman, R. Dynamic Programming. Princeton Univ. Press, 1957.

[68] Box, M.J. A new method of constrained optimization and a comparison with other methods. The Computer J., Vol. 8, (1965), pp. 42-52.

[69] ----. A comparison of several current optimization methods, and the use of transformations in constrained problems. The Computer J., Vol 9, (1966), pp. 67-77.

[70] Bracken, J., and McCormick, G.P. Selected Applications of Non-linear Programming. John Wiley & Sons, 1967.

[71] Colville, A.R. A comparative study of non-linear programming codes. International Symposium on Mathematical Programming, Princeton Univ., (Aug. 1967).

[72] Dorn, W.S. Non-linear programming -- a survey. Management Science, Vol. 9, No. 2, (Jan. 1963), pp.171-208.

[73] Duffin, R.J., Peterson, E.L., and Zener, C. Geometric Programming: Theory and Applications. John Wiley & Sons, 1967.

[74] Fletcher, R. Function minimization without evaluating derivatives -- a review. The Computer J., Vol. 8, (1965), pp. 33-41.

[75] Fletcher, R., and Reeves, C.M. Function minimization by conjugate gradients. The Computer J., Vol. 7, (1964), pp. 149-154.

[76] Goldfarb, D., and Lapidus, L. Conjugate gradient method for non-linear programming problems with linear constraints. I. & E.C. Fundamentals, Vol. 7, (1968), pp. 142-151.

[77] Hadley, G. Non-linear and Dynamic Programming. Addison-Wesley, 1964.

[78] Hartley, H.O. Non-linear programming by the simplex method. Econometrica, Vol. 29, (1961), pp. 223-237.

[79] Kunzi, H.P., Krelle, W., and Oettli, W. Non-linear Programming. Blaisdell, 1966.

[80] Mangasarian, O.L. Non-linear Programming. McGraw-Hill, 1969.

[81] Powell, M.J.D. An efficient method of finding the minimum of a function of several variables without calculating derivatives. The Computer J., Vol. 8, (1964), pp. 155-162.

[82] Rosen, J.B. The gradient projection method for non-linear programming. Part I. Linear constraints. SIAM J. on Applied Mathematics, Vol. 8, (1960), pp. 181-217.

[83] ----. The gradient projection method for non-linear programming. Part II. Non-linear constraints. SIAM J. on Applied Mathematics, Vol. 9, (1961), pp. 514-532.

[84] Scarf, H.E. The approximation of fixed points of a continuous mapping. SIAM J. on Applied Mathematics, Vol. 15, (1967), pp. 1328-1343.

[85] Spang, H.A. A review of minimization techniques for non-linear functions. SIAM Review, Vol. 4, (1962), pp. 343-365.

[86] Tucker, A.W. Linear and non-linear programming. Operations Research, Vol. 5, No. 2, (April 1957), pp. 244-257.

[87] Zangwill, W.I. Non-linear Programming -- A Unified Approach. Prentice-Hall, 1969.

[88] Zoutendijk, G. Non-linear programming: a numerical survey. SIAM J. on Control, Vol. 4, (1966), pp. 194-210.

D. Integer Programming

[89] Balas, E. Discrete programming by the filter method. Operations Research, Vol. 15, (1967), pp. 915-957.

[90] Balinski, M.L. Integer programming: methods, uses, computation. Management Science, Vol. 12, (1965), pp. 253-313.

[91] Beale, E.M.L. Survey of integer programming. Operational Research Quarterly, Vol. 16, (1965), pp. 219-228.

[92] Cabot, A.V., and Hurter, Jr., A.P. An approach to zero-one integer programming. Operations Research, Vol. 16, (1968), pp. 1206-1211.

[93] Dantzig, G.B. Discrete-variable extremum problems. Operations Research, Vol. 5, No. 2, (April 1957), pp. 266-277.

[94] Fox, B. Discrete optimization via marginal analysis. Management Science, Vol. 13, No. 3, (Nov. 1966), pp. 210-216.

[95] Freeman, R.J. Computational experience with a 'Balasian' integer programming algorithm. Operations Research, Vol. 14,(1966), pp. 935-941.

[96] Geoffrion, A.M. Integer programming by implicit enumeration and Balas' method. SIAM Review, Vol. 9, (1967), pp. 178-190.

[97] Gomory, R.E. On the relation between integer and non-integer solutions to linear programs. National Academy of Sciences, Vol. 53,(1965), pp. 260-265.

[98] Gross, O. A Class of Discrete-type Minimization Problems. RM-1644-PR, The Rand Corp., Santa Monica, Calif., (Feb. 1956).

[99] Graves, G.W., and Whinston, A. A new approach to discrete mathematical programming. Management Science, Vol. 15, (1968), pp. 177-190.

[100] Hu, T.C. Integer Programming and Network Flows. Addison-Wesley, 1969.

[101] Kaplan, S. Solution of the Lorie-Savage and similar integer programming problems by the generalized Lagrange multiplier method. Operations Research, Vol. 14, No. 6, (Nov-Dec. 1966), pp. 1130-1136.

[102] Land, A.H., and Doig, A.G. An automatic method of solving discrete programming problems. Econometrica, Vol. 28, (1960), pp. 497-520.

[103] Lawler, E.L., and Bell, M.D. A method for solving discrete optimization problems. Operations Research, Vol. 14, No. 6, (Nov-Dec. 1966), pp. 1098-1112.

[104] Markowitz, H.M., and Manne, A.S. On the solution of discrete programming problems. Econometrica, Vol. 25, (1957), pp. 84-110.

[105] Muth, J.F., and Thompson, G.L. Industrial Scheduling. Prentice-Hall, 1963.

[106] Nemhauser, G.L., and Ullmann, Z. A note on the generalized Lagrange multiplier solution to an integer programming problem. Operations Research, Vol. 16, (1968), pp. 450-453.

[107] Reiter, S., and Sherman, G. Discrete optimizing. SIAM J. on Applied Maths., Vol. 13, (1965), pp. 864-889.

[108] Tonge, F.M. A Heuristic Program for Assembly Line Balancing. Prentice-Hall, 1961.

[109] Watters, L.J. Reduction of integer polynomial programming problems to zero-one linear programming problems. Operations Research, Vol. 15,(1967), pp. 1171-1174.

[110] Weingartner, H.M. Mathematical Programming and the Analysis of Capital Budgeting Problems. Prentice-Hall, 1960.

[111] Young, R.D. A simplified primal (all-integer) integer programming algorithm. Operations Research, Vol.16, (1968), pp. 750-782.

B30036